UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Tomaž Dimnik

Spletna aplikacija za nadzor centralnega računalnika, operacijskih sistemov Linux in aplikacijskih strežnikov

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentorica: izr. prof. dr. Polona Oblak

Ljubljana, 2016

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Tomaž Dimnik,

z vpisno številko 63990175,

sem avtor diplomskega dela z naslovom:

Spletna aplikacija za nadzor centralnega računalnika, operacijskih sistemov Linux in aplikacijskih strežnikov.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Polone Oblak,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 8.6.2016 Podpis avtorja: _____

Zahvala

Naprej se zahvaljujem staršem, ki so mi študij omogočili in me podpirali skozi vsa leta mojega študija.
Zahvaljujem se mentorici izr. prof. dr. Poloni Oblak za podporo, pomoč in nasvete pri izdelavi diplomskega dela.
Zahvaljujem se tudi vsem sodelavcem, ki so mi v času študija pomagali.

Kazalo

Uporabljene kratice in simboli

PHP	PHP: Hypertext Preprocessor (skriptni programski jezik PHP)
HTML	HyperText Markup Language (označevalni jezik HTML)
AJAX	Asynchronous JavaScript and XML (skupina tehnik, uporabljena za ustvarjanje interaktivnih spletnih aplikacij)
JavaScript	programski jezik
CSS	Cascading Style Sheets (označevalni jezik, ki je v povezavi s HTML namenjen oblikovanju postavitve spletnih strani)
z/OS	64-bitni operacijski sistem za centralne računalnike
LDAP	Lightweight Directory Access Protocol (baza podatkov o uporabnikih, organizirana v drevesno strukturo)
TSM	Tivoli Storage Manager – IBM Spectrum Protect (sistem za varnostno kopiranje)
IBM	International Business Machines (največje računalniško podjetje na svetu)
WAS	WebSphere Application Server (aplikacijski strežnik WebSphere)
Perl	skriptni programski jezik Perl
Rexx	skriptni programski jezik Rexx
WLM	IBM Workload Manager (upravljalnik obremenitev na operacijskem sistemu z/OS)
USS	Unix System Services (okolje Unix znotraj operacijskega sistema z/OS)

CP	IBM System z Central Processor (splošni tip procesorja v okolju IBM System z)
zIIP	IBM System z Integrated Information Processor (procesor, namenjen obdelavam v podatkovnih bazah)
zAAP	IBM System z Application Assist Processor (procesor, namenjen obdelovanju aplikacij v javi, XML-u)
IFL	IBM System z Integrated Facility for Linux (procesor, namenjen delovanju operacijskega sistema Linux na centralnem računalniku)
DOM	Document Object Model (interakcija z elementi spletne strani preko JavaScripta)
HKOM	zasebno omrežje, ki je zasnovano za prenos podatkov med posameznimi zaključenimi celotami (FURS, ministrstva itd.)
RHEV	RedHat Enterprise Virtualization (platforma za virtualizacijo)
ZZZS	Zavod za zdravstveno zavarovanje Slovenije
MJU	Ministrstvo za javno upravo
z/VM	operacijski sistem na centralnem računalniku
AS	Address Space (virtualni naslovni prostor v operacijskem sistemu centralnega računalnika, individualno dodeljen vsaki aplikaciji, podobno kot procesi v okoljih z operacijskim sistemom Linux)
SPOOL	Simultaneous Peripheral Operations On-Line (repozitorij za vse vhodne posle in večino izhodnih poslov na centralnem računalniku)
RAM	Random Access Memory (pomnilnik z naključnim dostopom)
RACF	IBM Resource Access Control Facility (varnostni sistem, ki zagotavlja nadzor dostopa do operacijskega sistema centralnega računalnika)

Povzetek

V delu predstavimo idejo in realizacijo spletne aplikacije za nadzor delovanja centralnega računalnika, strežnikov z operacijskim sistemom Linux in aplikacijskih strežnikov. Spletna aplikacija je namenjena skrbnikom omenjenih sistemov kot pomoč za lažje razumevanje trenutnega stanja, obremenitev in delovanja posameznih komponent sistemov.

Spletna aplikacija je razvita v programskem jeziku PHP z uporabo orodja Notepad++. Kot pomoč pri predstavitvi podatkov so uporabljeni tudi programski jeziki HTML, CSS, JavaScript in AJAX. Za podatke in njihovo shranjevanje skrbi podatkovni strežnik MariaDB.

Ključne besede:

spletna aplikacija, nadzor sistemov, predstavitev podatkov, PHP, uporabniški vmesnik, aplikacijski strežnik, centralni računalnik, operacijski sistem Linux.

Abstract

This work presents the idea and the realization of web application for monitoring the operation of the mainframe computer, servers with Linux operating system and application servers. Web application is intended for administrators of these systems, as an aid to better understand the current state, load and operation of the individual components of the server systems.

The web application is developed in the programming language PHP using tool Notepad ++. Programming languages HTML, CSS, JavaScript and AJAX are used to assist in the presentation of data. MariaDB database server is used for data storage.

Key words:

Web application, system monitoring, data presentation, PHP, user interface, application server, mainframe computer, Linux operating system.

1 Uvod

V današnjem svetu si želimo vse več nadzora nad dejavniki, ki bi lahko povzročili izpad ali omejeno delovanje računalniških in strežniških sistemov, aplikacij, aplikacijskih strežnikov in drugih sistemskih komponent, ki zagotavljajo nemoteno delovanje aplikacij na spletu. Spletne aplikacije v današnjih časih so v trendu naraščanja in so postale osnova za večino uradniških storitev javne uprave, sodnega sistema, finančne uprave, notranjih zadev in podobnih javnih ustanov, ki jih vsi uporabljamo. Dejstvo je, da je treba pri takih aplikacijah zagotoviti čim krajše odzivne čase aplikacij in čim krajši čas nedelovanja aplikacij.

Naloga sistemski skrbnikov je, da pred končnim uporabnikom ugotovijo pojavitev težav, zaradi katerih bi lahko prišlo do delnega ali celo popolnega izpada delovanja neke aplikacije. Za dani problem obstaja več rešitev. Sprotno spremljanja sistemov pri velikem številu virtualnih ali realnih strežnikov ne pride v poštev. Obstaja pa večje število komercialnih ali odprtokodnih aplikacij, namenjenih nadzoru delovanja sistemov, vendar so te večinoma omejene na spremljanje operacijskih sistemov Windows in Linux.

V podjetju, ki se ukvarja s sistemsko in operativno podporo centralnih računalnikov, strežnikov z operacijskim sistemom Linux in podatkovnih baz, se je zaradi različnih tipov sistemskih okolij pojavila težava, kako spremljati vse sisteme hkrati in zagotoviti optimalno delovanje vseh aplikacij. Na voljo je bila uporaba različnih odprtokodnih nadzornih sistemov, kot sta na primer Nagios in Zabbix, ki pa sta namenjena nadzoru operacijskih sistemov Linux in Microsoft Windows. Hkrati se je pojavila potreba po nadzoru nad zelo specifičnim operacijskim sistemom z/OS, ki teče na centralnem računalniku. Ker tako specifična aplikacija, ki bi zagotavljala spremljanje in nadzor vseh potrebnih komponent različnih sistemov, ne obstaja, se je porodila ideja za razvoj spletne aplikacije, ki vključuje spremljanje in nadzor več različnih operacijskih sistemov in aplikacijskih strežnikov.

Namen diplomskega dela je izdelava in prikaz spletne aplikacije za nadzor centralnega računalnika, strežnikov z operacijskim sistemom Linux in aplikacijskih strežnikov, s katero lahko uporabnik na enostaven način ugotovi trenutne težave na sistemih, kontrolira delovanje aplikacijskih strežnikov, primerno ukrepa in s tem optimizira odzivne čase ter prepreči delni ali popolni izpad delovanja aplikacij. Podatki, imena in primeri, predstavljeni v tem diplomskem delu, bodo izmišljeni, saj zaradi tajnosti podatkov pravih podatkov ne smemo razkrivati.

Diplomsko delo je organizirano na naslednji način.

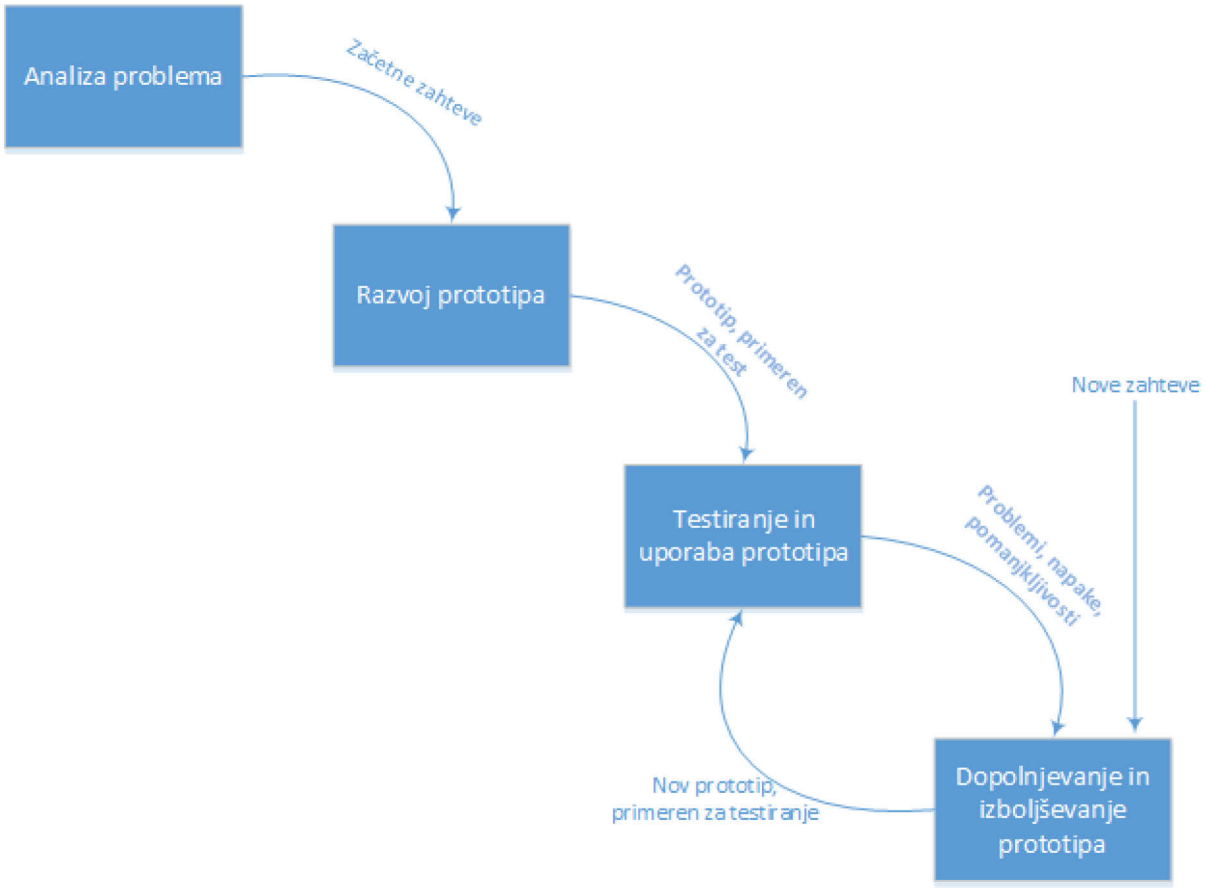
V drugem poglavju opišemo načrtovanje in izdelavo aplikacije. Predstavljene so tudi tehnologije in orodja, uporabljena pri izdelavi.
V tretjem poglavju opišemo uporabniški vmesnik, vstopno stran in glavni portal.
Nadzor centralnega računalnika opišemo v četrtem poglavju. Predstavimo centralni računalnik in opišemo parametre, ki jih spremljamo.
Nadzor strežnikov z operacijskim sistemom Linux opišemo v petem poglavju. Predstavimo in opišemo parametre, ki jih spremljamo.
V šestem poglavju opišemo nadzorno ploščo za aplikacijske strežnike. Predstavimo rešitev, s katero enostavno kontroliramo stanje in obnašanje aplikacijskih strežnikov.
V sedmem poglavju povzamemo sklepne ugotovitve.

2 Načrtovanje in razvoj

2.1 Prototipni model razvoja

Pri razvoju aplikacije je poleg začetnih zahtev težko določiti, kakšen naj bi bil uporabniški vmesnik med aplikacijo in uporabnikom. V takšnih primerih je koristno uporabiti prototipni model razvoja (slika 1), ki temelji na izdelavi prototipov in njihovih izboljšavah tako dolgo, dokler ni izdelan prototip, ki ustreza vsem zahtevam uporabnika. Uporaba prototipov olajša komunikacijo z uporabnikom, zato jih uporabljamo v različnih fazah razvoja. Izdelava dobrega prototipa, je v veliki meri odvisna od sodelovanja med razvijalcem in uporabnikom. Ko dosežeta dogovor glede zahtev sistema, razvijalec razvije prototip. Uporabnik dobi prototip v uporabo oziroma testiranje in razvijalcu sporoči pomanjkljivosti (težave, napake, nove zahteve) v prototipu. Razvijalec na podlagi uporabnikovih povratnih informacij izdela novo verzijo prototipa in ga preda uporabniku v uporabo in testiranje. Postopek se ponavlja tako dolgo, dokler uporabnik s prototipom ni zadovoljen. Prototipi se lahko uporabljajo le kot majhen del zbiranja specifikacij sistema, to je z namenom, da uporabnik vidi približno sliko bodočega sistema. Potem se zavržejo (throwaway or close-ended prototyping). Lahko pa se prototipi uporabljajo tudi kot osnova za izdelavo produkcijskega sistema (evolutionary or breadboard prototyping). Pri tem načinu je glavni cilj narediti zelo robusten model prototipa, ki ga nenehno dopolnjujemo in izboljšujemo. V tem primeru je prototip jedro sistema. To jedro potem izpopolnjujemo in izboljšujemo, dokler se ne razvije v končno aplikacijo. V našem primeru smo uporabili evolucijsko prototipiranje, saj je osnovni prototipni model služil kot osnova za končno verzijo aplikacije.

S prototipnim modelom razvoja smo dosegli bolj izpopolnjeno končno verzijo aplikacije, večje zadovoljstvo uporabnika, saj je aktivno sodeloval pri dopolnjevanju in popravkih aplikacije, najbolj pomembno pa je, da je bil čas razvoja aplikacije bistveno krajši.



Slika 1: Prototipni razvoj

Spletna aplikacija je bila razvita v trinivojski arhitekturi, ki jo sestavljajo:

- podatkovni del,
- aplikacijski del,
- predstavitveni del.

2.2 Analiza problema in opis aplikacije

Osnovni problem, o katerem govorimo, je nadziranje več različnih tipov sistemskih okolij in s tem zagotavljanje optimalnega delovanja vseh aplikacij, in to znotraj ene same spletne aplikacije za nadzor.

Pri analizi takega problema je bilo prvo vprašanje, ali taka aplikacije že obstaja, ali se je kakšno sorodno podjetje v Sloveniji ali v tujini že srečalo s podobnim problemom. Takih podjetij, ki bi hkrati uporabljala aplikacijske strežnike, podatkovne baze, strežnike LDAP, spletne strežnike na več tako specifično različnih sistemskih okoljih (z/OS, z/VM, Linux, Windows), po našem poznavanju področja ni veliko. Centralni računalnik imajo naslednja

večja podjetja oz. ustanove v Sloveniji: Petrol, ZZZS, NLB, Informatika, MJU. Pri teh podjetjih oz. ustanovah so osredotočeni na centralni računalnik. Velika večina produktov teče na operacijskem sistemu z/OS ali z/VM. Temu primerno uporabljajo komercialne produkte za nadzor sistemov centralnega računalnika (npr. družino produktov IBM Tivoli Omegamon XE, RMF Data Portal).
V tujini pa je organiziranost delovanja sistemskih skrbnikov drugačna. Po priporočilih podjetja IBM mora za centralni računalnik skrbeti večje število primerno izobraženih sistemskih skrbnikov, ki so ozko usmerjeni v točno določen produkt. Vsak izmed njih skrbi za določen del, katerega nadzirajo večinoma z družino produktov IBM Tivoli za nadzor centralnega računalnika. Podoben produkt smo testirali tudi pri nas, vendar se uporabnik zaradi ekstremno visoke cene zanj ni odločil. Tako se je porodila ideja za razvoj spletne aplikacije, ki vključuje spremljanje in nadzor več različnih operacijskih sistemov in aplikacijskih strežnikov.

Spletna aplikacija je namenjena nadzoru centralnega računalnika, operacijskih sistemov Linux in aplikacijskih strežnikov. Uporabniku omogoča, da na enostaven način ugotavlja trenutne težave na sistemih, kontrolira delovanje aplikacijskih strežnikov, primerno ukrepa in s tem optimizira odzivne čase ter prepreči delni ali popolni izpad delovanja aplikacij. Aplikacija je poenostavljena do te mere, da uporabnik za uporabo potrebuje le spletni brskalnik in internetno povezavo.

Spletna aplikacija je razdeljena na tri vsebinske sklope:

- nadzor centralnega računalnika (dostopen samo prijavljenim obiskovalcem): vsebuje podatke, pomembne za nadzor in ugotavljanje delovanja ključnih komponent centralnega računalnika, aplikacijskih strežnikov, podatkovnih baz, strežnikov LDAP, strežnika TSM itd.;
- nadzor virtualnih in realnih sistemov z operacijskem sistemom Linux (dostopen samo prijavljenim uporabnikom): vsebuje informacije, pomembne za nadzor in ugotavljanje delovanja ključnih komponent operacijskih sistemov Linux, aplikacijskih strežnikov, varnostnih kopij, strežnikov Lotus Domino itd.;
- nadzor aplikacijskih strežnikov IBM Websphere (dostopen samo prijavljenim uporabnikom): s pomočjo nadzorne plošče lahko ugotavljamo trenutno stanje aplikacijskega strežnika, omogoča nam tudi izklop, vklop in brezpogojni izklop posameznih delov aplikacijskega strežnika.

2.2.1 Analiza zahtev

Pri analizi zahtev smo ugotovili, da mora spletna aplikacija ustrezati naslednjim zahtevam:

- za delovanje sta potrebna samo spletni brskalnik in internetna povezava,
- poenostavljena mora biti do te mere, da je uporabniku dovolj le znanje o uporabi spletnega brskalnika,
- zgrajena mora biti tako, da omogoča čim enostavnejšo dopolnjevanje, dodajanje in odstranjevanje komponent,
- delovati mora v vseh novejših spletnih brskalnikih (Internet Explorer, Chrome, Opera, Mozilla Firefox, Safari),
- zgrajena mora biti tako, da omogoča delo več uporabnikom hkrati.

2.3 Načrtovanje aplikacije

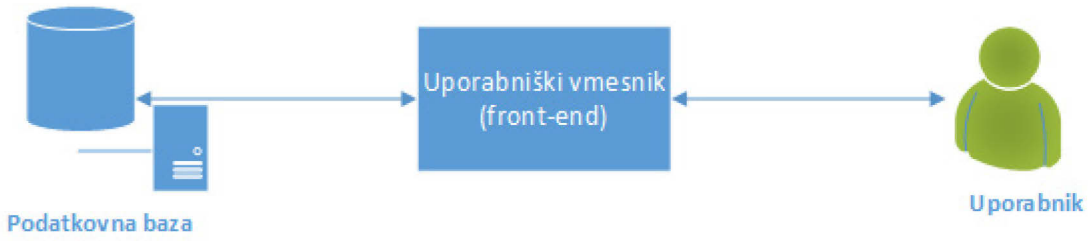
V tem poglavju so predstavljeni zgradba, način delovanja in podatkovni model aplikacije.

2.3.1 Zgradba in način delovanja

Aplikacija je sestavljena iz treh glavnih komponent:

- uporabniški vmesnik (front-end),
- aplikacijski del (back-end),
- nadzorna plošča (nadzor aplikacijskih strežnikov IBM WAS).

Uporabniški vmesnik je sestavljen iz vstopne strani, ki omogoča dostop samo registriranim uporabnikom, in iz predstavitvenega portala. Predstavitveni portal se deli na zavihke, ki predstavljajo nadzor nad sistemi, in zavihek, v katerem deluje nadzorna plošča, namenjena upravljanju aplikacijskih strežnikov IBM WAS. Del predstavitvenega portala, v katerem spremljamo delovanje sistemov, je narejen na osnovi prosto dostopne kode »RSS boxes« po licenci LGPL. Uporabljen je predvsem zaradi funkcionalnosti osveževanja in omogoča nastavitev različnih osveževalnih časov za različne komponente (okna v predstavitvenem delu). Koda je popravljena in dodelana do te mere, da omogoča predstavitev podatkov, pomembnih za nadzor sistemov. Naloga uporabniškega vmesnika je pridobivanje podatkov iz podatkovne baze, njihova obdelava in prikaz v spletni aplikaciji (slika 2).



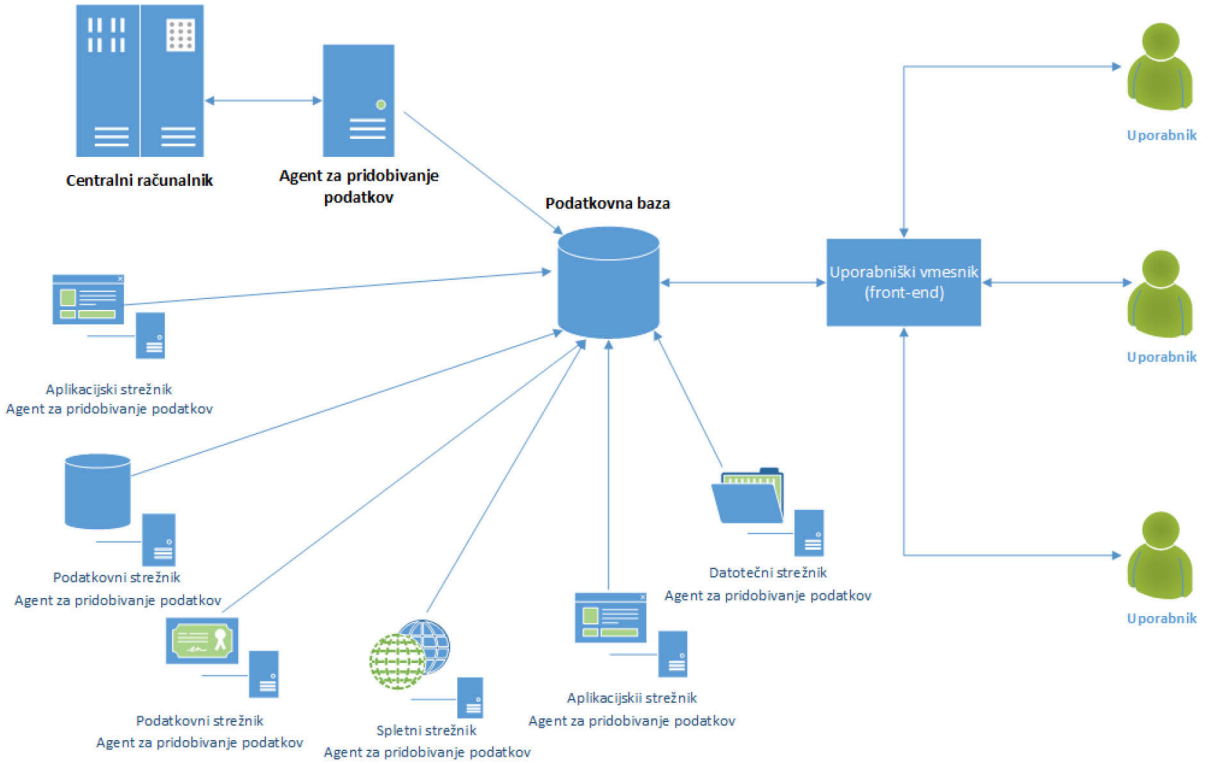
Slika 2: Delovanje uporabniškega vmesnika

Glavni del spletne aplikacije predstavlja skupek agentov, katerih naloga je, da v določenih časovnih intervalih pridobivajo informacije o porabi, stanju, delovanju, nedelovanju različnih sistemskih komponent na vsakem sistemu, ki ga želimo nadzirati (slika 3).
Ločimo dve vrsti agentov:

- agent, namenjen spremljanju centralnega računalnika z operacijskim sistemom z/OS,
- agent, namenjen spremljanju virtualnih in realnih strežnikov z operacijskim sistemom Linux.

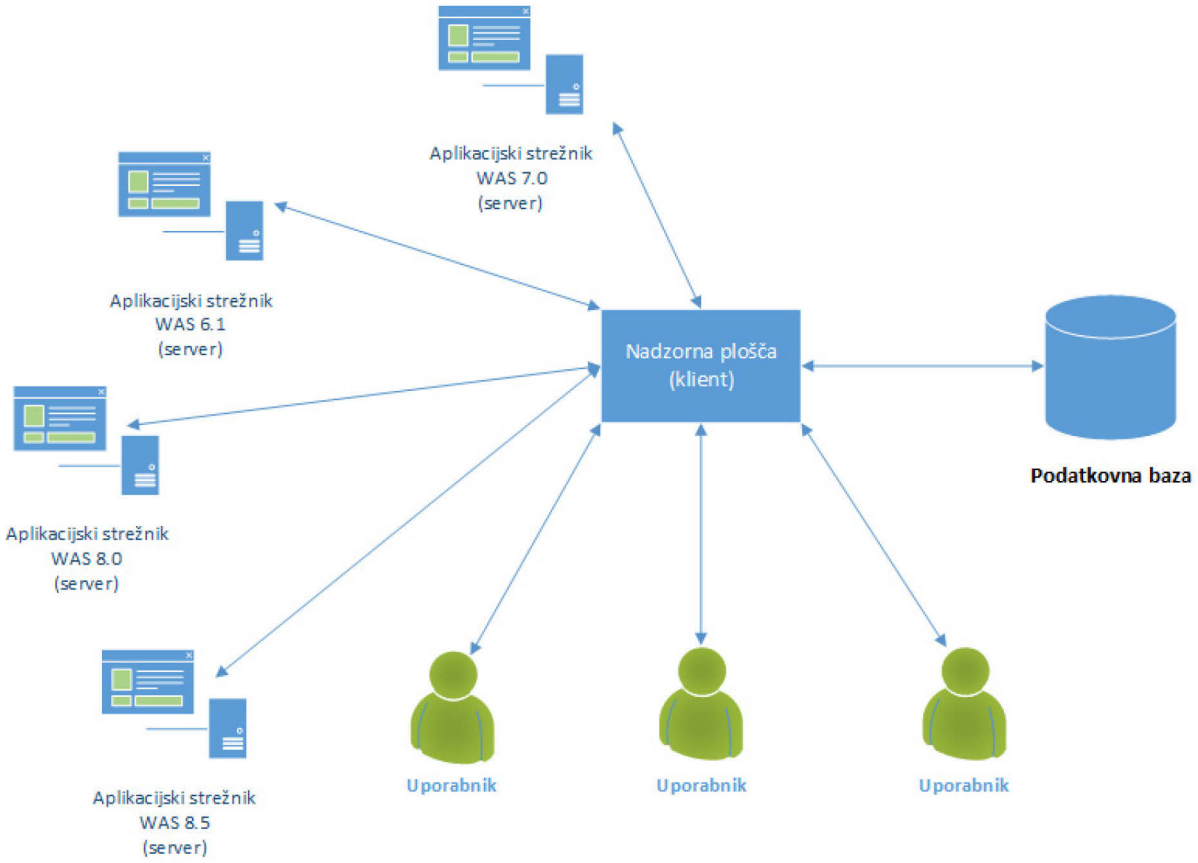
Agent, namenjen spremljanju centralnega računalnika, je samostojna aplikacija PHP, katera za delovanje potrebuje okolje PHP na kateremkoli računalniku z mrežno povezavo do centralnega računalnika. Njen namen je v določenih intervalih zbirati informacije o virih, stanju, delovanju in nedelovanju različnih komponent centralnega računalnika. Del podatkov se pridobiva s pomočjo »RMF Data Portal for z/OS«. To je aplikacija, namenjena spremljanju delovanja centralnega računalnika. Ostali del podatkov pa se pridobiva s klicem različnih skript v skriptnih jezikih Perl in Rexx preko spletnega strežnika, ki deluje na centralnem računalniku. Podatki se pridobivajo v različnih časovnih intervalih glede na potrebo. Ti podatki se nato obdelajo in zapišejo v podatkovno bazo. Pri tem je treba paziti, da s prenizkimi časovnimi intervali pridobivanja podatkov ne obremenjujemo preveč centralnega računalnika.

Agent, namenjen spremljanju strežnikov z operacijskim sistemom Linux, je samostojna aplikacija PHP, ki za svoje delovanje potrebuje okolje PHP. Agent, namenjen spremljanju strežnikov z operacijskim sistemom Linux, deluje na strežniku, ki ga nadzira. Deluje tako, da izvaja določene ukaze, s katerimi pridobiva podatke o sistemu, te podatke nato obdela in zapiše v podatkovno bazo. Tudi pri tem agentu je treba paziti na intervale pridobivanja podatkov, da dodatno ne obremenjujemo strežnika, ki ga nadzorujemo.



Slika 3: Zgradba in delovanje spletne aplikacije

Nadzorna plošča za aplikacijske strežnike IBM WAS je namenjena nadzoru delovanja aplikacijskih strežnikov tipa IBM WAS. S pomočjo nadzorne plošče lahko ugotavljamo trenutno stanje aplikacijskega strežnika, omogoča nam tudi izklop, vklop in brezpogojni izklop posameznih delov aplikacijskega strežnika. Nadzorna plošča deluje po modelu klient – strežnik (slika 4). Na vsakem strežniku, na katerem deluje aplikacijski strežnik tipa IBM WAS, deluje tudi samostojna aplikacija PHP, ki posluša na določenih vratih in se odziva na ukaze, ki jih pošiljamo iz nadzorne plošče. Aplikacijske strežnike lahko nadziramo v okoljih z operacijskim sistemom Windows ali Linux, temu primerno so spisane skripte za zagon, ustavitev, preverjanje stanja.



Slika 4: Delovanje nadzorne plošče

2.3.2 Podatkovni model

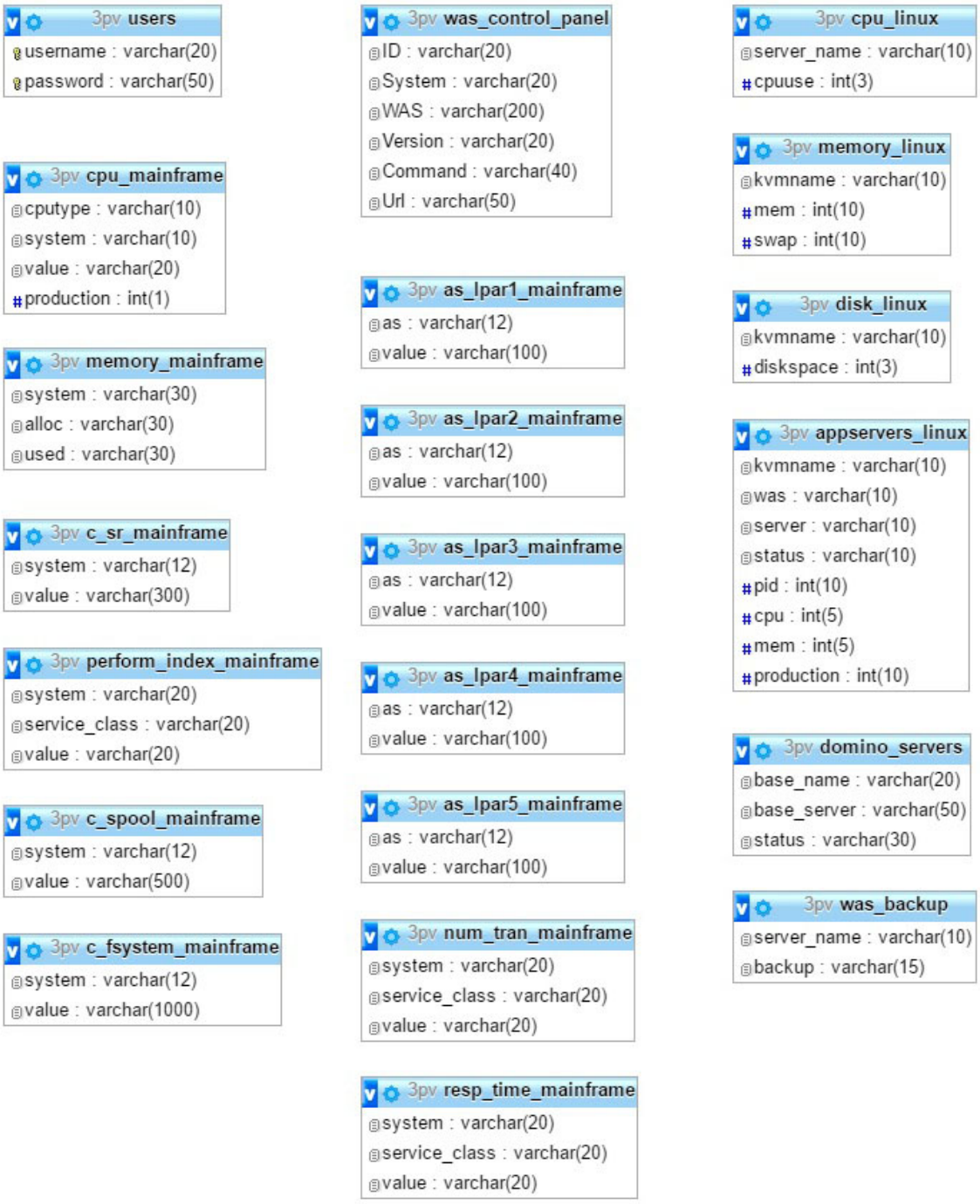
Za načrtovanje spletne aplikacije je bil uporabljen podatkovni model, ki je predstavljen na sliki 5.

Podatkovni model vsebuje naslednje entitete:

- USERS: vsebuje osnovne podatke o registriranem uporabniku
- DOMINO_SERVERS: podatki o delovanja strežnikov Lotus Domino
- WAS_CONTROL_PANEL: podatki, povezani z nadzorno ploščo
- APPSERVERS_LINUX: podatki o aplikacijskih strežnikih, ki delujejo v okolju Linux
- CPU_LINUX: podatki o trenutni porabi procesorja v okoljih Linux
- MEMORY_LINUX: podatki o trenutni porabi glavnega pomnilnika v okoljih Linux
- DISK_LINUX: podatki o trenutni zasedenosti diskov v okoljih Linux
- WAS_BACKUP: podatki o stanju varnostnih kopij WAS-aplikacijskih strežnikov
- CPU_MAINFRAME: podatki o trenutni porabi procesorjev v okolju z/OS
- MEMORY_MAINFRAME: podatki o trenutni porabi glavnega pomnilnika v okolju z/OS

- PERFORM_INDEX_MAINFRAME: podatki o performančnih indeksih WLM-ja po posameznih servisnih klasah
- NUM_TRAN_MAINFRAME: podatki o številu transakcij na sekundo po posameznih servisnih klasah
- RESP_TIME_MAINFRAME: podatki o povprečnih odzivnih časih po posameznih servisnih klasah
- C_FSYSTEM_MAINFRAME: podatki o trenutni zasedenosti diskov v USS v okolju z/OS
- C_SR_MAINFRAME: podatki o sistemskih zahtevah v okolju z/OS
- C_SPOOL_MAINFRAME: podatki o trenutni zasedenosti datoteke SPOOL
- AS_LPAR1_MAINFRAME: podatki o programih, ki tečejo na particiji LPAR1 v okolju z/OS
- AS_LPAR2_MAINFRAME: podatki o programih, ki tečejo na particiji LPAR2 v okolju z/OS
- AS_LPAR3_MAINFRAME: podatki o programih, ki tečejo na particiji LPAR3 v okolju z/OS
- AS_LPAR4_MAINFRAME: podatki o programih, ki tečejo na particiji LPAR4 v okolju z/OS
- AS_LPAR5_MAINFRAME: podatki o programih, ki tečejo na particiji LPAR5 v okolju z/OS

Na podlagi podatkovnega modela so bile ustvarjene tabele v podatkovni bazi MariaDB.



Slika 5: Podatkovni model spletne aplikacije

Relacij med tabelami nismo uporabili, saj tabele vsebujejo malo podatkov, ki se v kratkih časovnih intervalih stalno prepisujejo. Arhivskih podatkov ni, zato z uporabo relacij med tabelami ne bi pridobili kakšnih opaznih razlik.

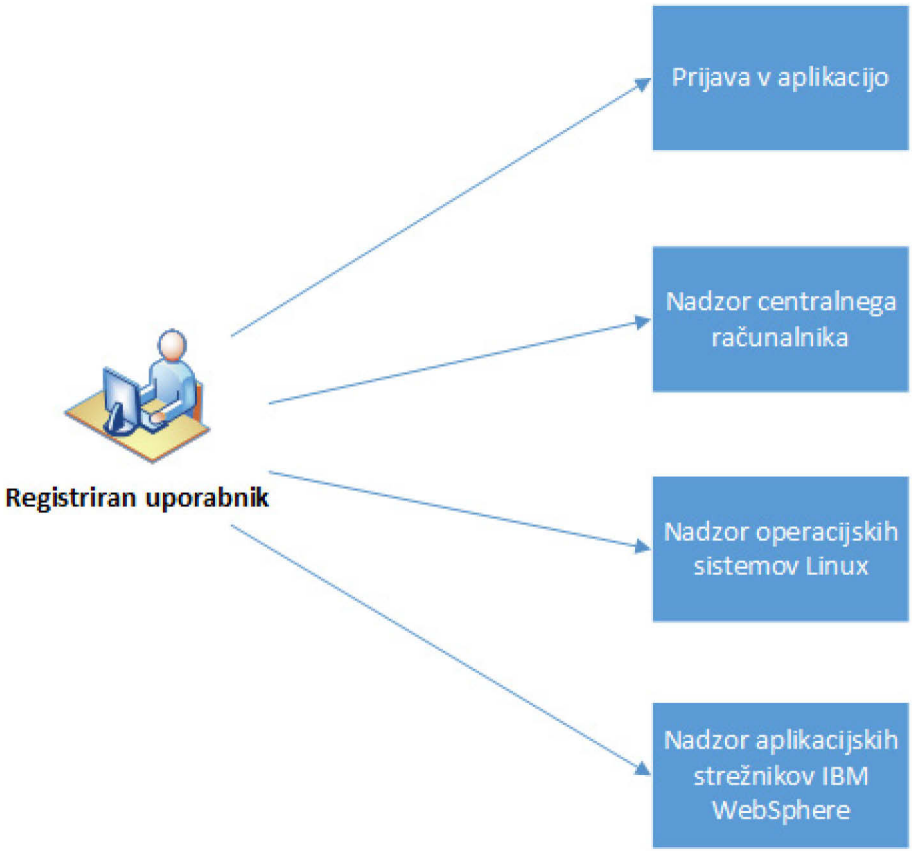
2.3.3 Primeri uporabe spletne aplikacije

Z namenom prikaza možnosti uporabe spletne aplikacije je bil narejen diagram primerov uporabe. Diagram primerov uporabe nam prikazuje, kako lahko registriran uporabnik uporablja spletno aplikacijo (slika 6). Dostop do spletne aplikacije neregistriranim uporabnikom ni dovoljen.

Primeri uporabe spletne aplikacije za registriranega uporabnika so naslednji.

- Prijava v aplikacijo: registriran uporabnik se prijavi v spletno aplikacijo z uporabniškim imenom in geslom.
- Nadzor delovanja centralnega računalnika: uporabnik nadzira delovanje različnih komponent centralnega računalnika. Sistem omogoča nadzor naslednjih komponent:
    - spremljanje zasedenosti različnih tipov procesorjev po posameznih logičnih particijah (CP, zIIP, zAAP, IFL)
    - spremljanje zasedenosti glavnega pomnilnika po posameznih logičnih particijah
    - spremljanje zasedenosti diskovnih kapacitet
    - spremljanje števila transakcij na sekundo po posameznih servisnih klasah
    - spremljanje povprečnega odzivnega časa po posameznih servisnih klasah
    - spremljanje performančnega indeksa WLM-ja po posameznih servisnih klasah
    - spremljanje delovanja posameznih particij, kar vključuje: zasedenost diskov, spremljanje sistemskih zahtev, zasedenost datoteke SPOOL, delovanje spletnih strežnikov, aplikacijskih strežnikov, podatkovnih baz, strežnikov LDAP, sistemskih procesov itd.
    - spremljanje izvajanja varnostnih kopij aplikacijskih strežnikov
    - spremljanje delovanja sistema za varnostno kopiranje (TSM)
- Nadzor delovanja strežnikov z operacijskim sistemov Linux: uporabnik nadzira delovanje različnih komponent strežnikov z operacijskih sistemom Linux. Sistem omogoča nadzor naslednjih komponent:
    - spremljanje zasedenosti procesorjev
    - spremljanje zasedenosti glavnega pomnilnika
    - spremljanje zasedenosti diskovnih kapacitet
    - spremljanje izvajanja varnostnih kopij aplikacijskih strežnikov
    - spremljanje delovanja domino strežnikov
    - spremljanje delovanja aplikacijskih strežnikov (s prikazom porabe pomnilnika in procesorja za vsak posamezen proces)
- Nadzor aplikacijskih strežnikov tipa WebSphere: uporabnik s pomočjo nadzorne plošče spremlja delovanje aplikacijskega strežnika po posameznih komponentah

(deployment manager, node, server1, server2, server3 ...). Nadzorna plošča uporabniku omogoča status delovanja, izklop, vklop in brezpogojen izklop vsake komponente aplikacijskega strežnika posebej.



Slika 6: Diagram primerov uporabe

2.4 Uporabljena orodja in tehnologije

V tem poglavju opišemo orodja in tehnologije, ki so bila uporabljena pri razvoju, načrtovanju in izdelavi spletne aplikacije.

2.4.1 Orodja

Orodje, ki je bilo uporabljeno za razvoj, se imenuje Notepad++ [10], ki je preprost in učinkovit urejevalnik izvorne kode.

Za upravljanje s podatkovno bazo, dodajanje, brisanje tabel, entitet in atributov je bilo uporabljeno orodje phpMyAdmin [12].

2.4.2 Tehnologije

PHP

PHP Hypertext Preprocessor [11] je razširjen odprtokodni programski jezik za izdelavo dinamičnih spletnih strani. Programski jezik je bil napisan kot skupek programov v programskem jeziku C. Napisal ga je dansko-kanadski programer Rasmus Lerdorf leta 1994. Programski jezik je primerljiv s programskimi jeziki C, Perl, VBScript, ASP in podobnimi.
V nasprotju s HTML-stranmi strežnik PHP ne pošlje skripte brskalniku uporabnika, temveč jo izvede na strani strežnika in šele nato pošlje rezultat (HTML) brskalniku uporabnika. PHP se izvaja na spletnem strežniku in ne na strani uporabnika kot na primer JavaScript, HTML, CSS in podobni. PHP je enostaven in neodvisen od okolja, v kateremu deluje (Windows, Linux, z/OS).

AJAX

AJAX [1] (asinhroni JavaScript in XML) je nabor medsebojno povezanih tehnologij, ki se uporabljajo pri razvoju interaktivnih spletnih aplikacij. Bistvena prednost aplikacij AJAX je, da si izmenjujejo podatke s strežnikom asinhrono v ozadju, kar nam omogoča osveževanje samo dela spletne strani, medtem ko uporabnik lahko nemoteno uporablja spletno stran.

jQuery

jQuery [7] je majhna in hitra knjižnica JavaScript z bogatim naborom naprednih učinkov. Z vidika uporabe je ena izmed najbolj uporabljenih tovrstnih knjižnic na svetu. Omogoča lažjo navigacijo po dokumentih, izbiro elementov DOM, izdelavo animacij, manipulacijo z dogodki in razvoj aplikacij AJAX.

JavaScript

JavaScript [6] je objektni skriptni jezik, namenjen ustvarjanju interaktivnih spletnih strani. JavaScript sodeluje s spletno stranjo in s tem poživi spletno stran z dinamičnim izvajanjem. Skriptni jezik podpirajo vsi novejši spletni brskalniki.

MariaDB

MariaDB je sistem za upravljanje s podatkovnimi zbirkami. Je odprtokodna implementacija relacijske podatkovne zbirke. Za delo s podatki uporablja jezik SQL. MariaDB deluje na principu odjemalec – strežnik. Obstaja veliko število odjemalcev in zbirk ukazov za dostop do podatkovne zbirke (Database Workbench, DBEdit, HeidiSQL, Navicat, phpMyAdmin, SQLyog).

Spletni strežnik Apache

Apache je odprtokodni spletni strežnik, ki je igral ključno vlogo pri širjenju spleta. Je najbolj uporabljan spletni strežnik na svetu. Gosti približno 48 % vseh svetovnih spletnih strani.

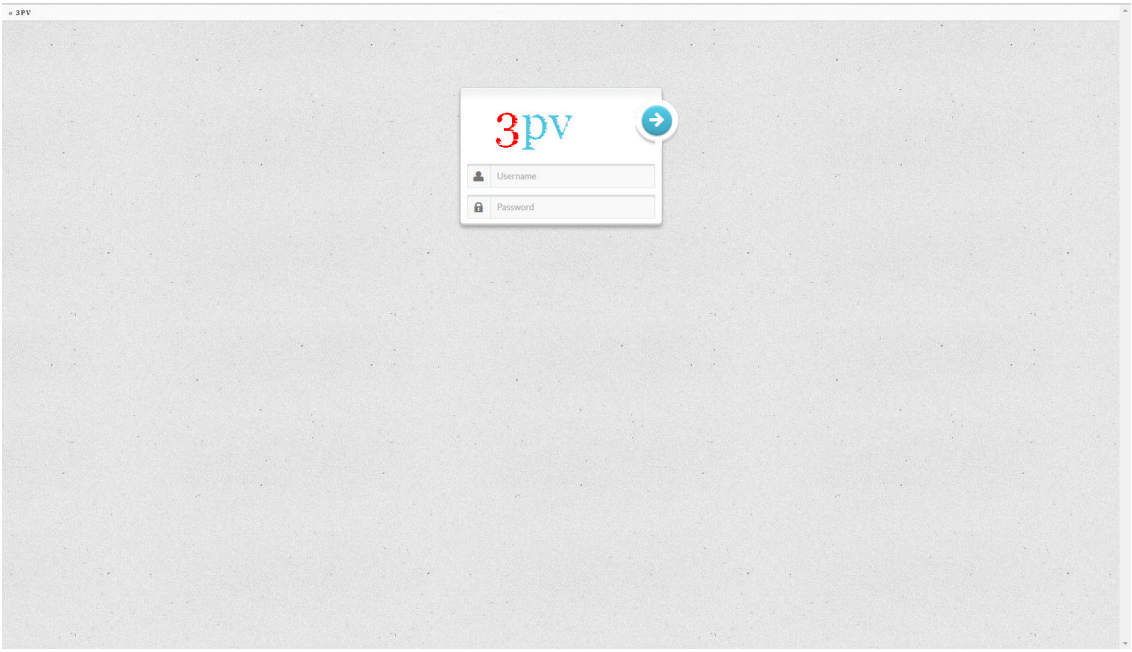
3 Uporabniški vmesnik

V tem poglavju sta predstavljena vstopna stran in glavni portal spletne aplikacije.

3.1 Vstopna stran

Vstopna stran se prikaže obiskovalcu ob prihodu na spletni naslov, ki gosti spletno aplikacijo. Vstopna stran (slika 7) omogoča uporabniku prijavo v portal spletne aplikacije. Zaradi tajnosti podatkov je dostop omogočen samo pooblaščenim uporabnikom. Glavni namen vstopne strani in njenih komponent je prav onemogočanje dostopa nepooblaščenim uporabnikom skozi celoten portal spletne aplikacije.

Pooblaščeni uporabnik se prijavi v spletno aplikacijo z uporabniškim imenom in geslom. Za kriptiranje gesla se uporablja algoritem MD5. Ker spletna aplikacija deluje znotraj omrežja HKOM, potrebe po večji varnosti ni, saj je dostop do omrežja HKOM možen od znotraj, preko fizično priklopljenih računalnikov ali od zunaj z RSA-kodirnim ključem preko omrežja Cisco VPN.



Slika 7: Spletna aplikacija (vstopna stran)

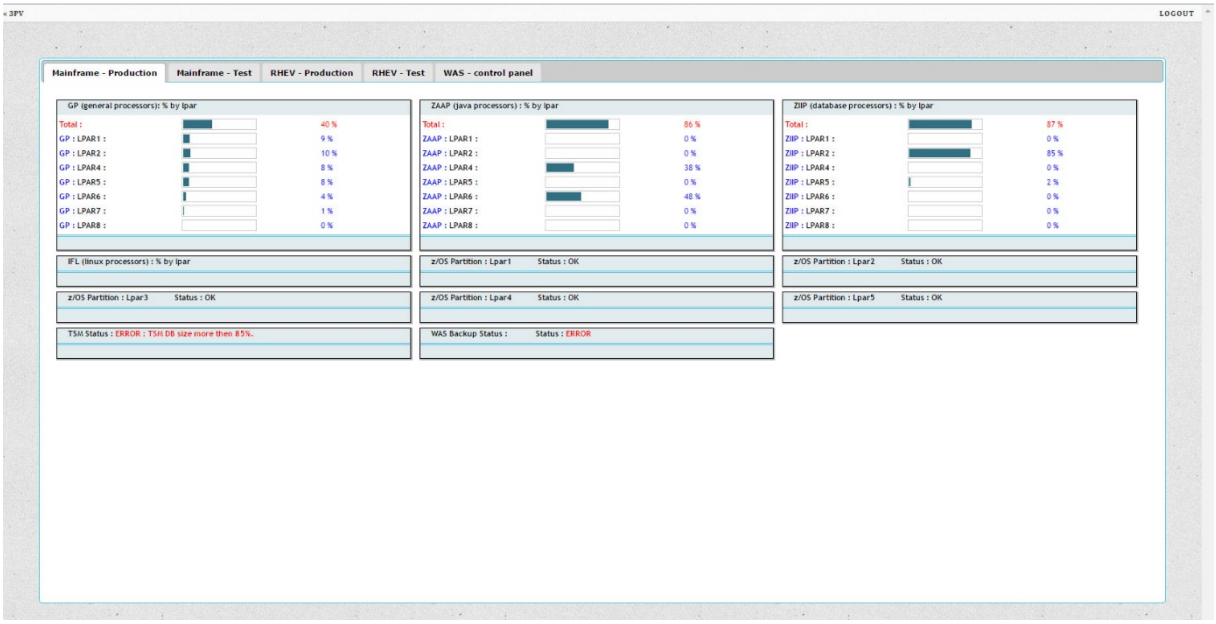
3.2 Glavni portal

Glavni portal spletne aplikacije (slika 8) se prikaže obiskovalcu po uspešni prijavi v spletno aplikacijo. Glavna naloga portala je predstavitev podatkov, ki jih nadziramo. Na vseh podstraneh glavnega portala je levo zgoraj ime spletne aplikacije, desno zgoraj pa povezava »LOGOUT«, namenjena odjavi iz spletne aplikacije.

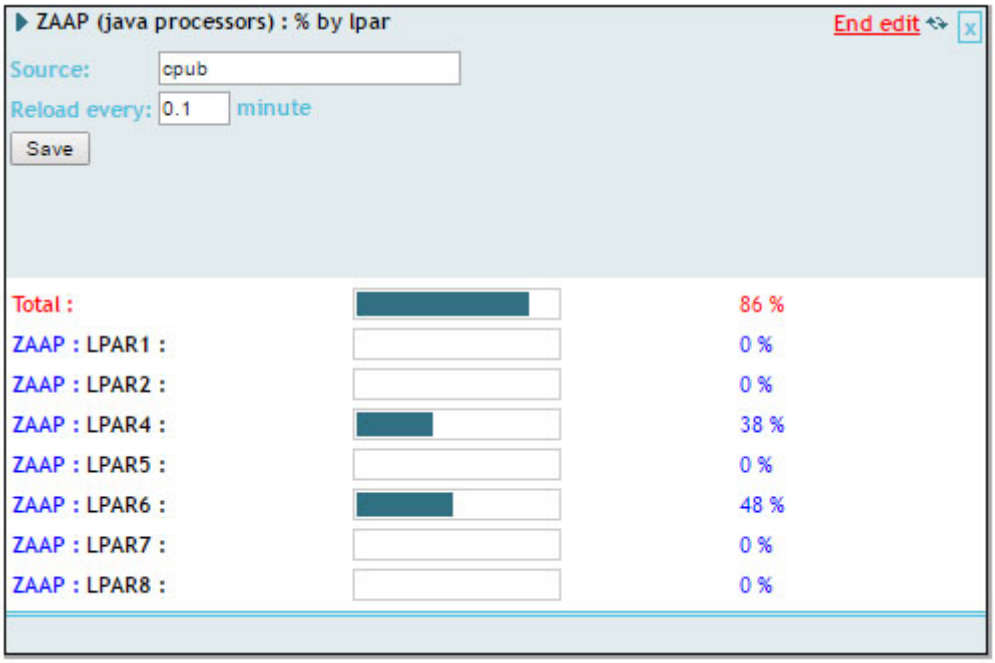
Pod glavo portala je nameščena glavna navigacija. Glavna navigacija obsega naslednje postavke:

- Mainframe – Production (predstavitev produkcijskih podatkov, namenjenih nadzoru centralnega računalnika)
- Mainframe – Test (predstavitev testnih podatkov, namenjenih nadzoru centralnega računalnika)
- RHEV – Production (predstavitev produkcijskih podatkov, namenjenih nadzoru virtualnih strežnikov tipa RHEV)
- RHEV – Test (predstavitev testnih podatkov, namenjenih nadzoru virtualnih strežnikov tipa RHEV)
- WAS – Control Panel (nadzorna plošča, namenjena spremljanju delovanja aplikacijskih strežnikov WAS)

Znotraj vsake postavke se nahajajo dinamične vsebine glavnega portala. Dinamične vsebine so predstavljene z okni, katerih vsebina predstavlja nadzor določenega sklopa znotraj postavke (slika 8). V vsakemu oknu se vsebina osvežuje v določenem intervalu, ostali del spletne aplikacije pa se ne spreminja. Čas osveževanja je mogoče spreminjati znotraj vsakega okna, torej za vsak sklop, ki ga nadziramo (slika 9).



Slika 8: Glavni portal spletne aplikacije



Slika 9: Prikaz nastavitve osveževanja za določen sklop

4 Nadzor delovanja centralnega računalnika

V tem poglavju so predstavljeni centralni ali osrednji računalnik IBM (IBM Mainframe) [5, 9, 15] in parametri, ki jih na centralnem računalniku spremljamo, njihov pomen in kako ukrepati v primeru, da njihove vrednosti niso v mejah optimalnega delovanja.

4.1 Predstavitev centralnega računalnika IBM

Beseda »mainframe« ali centralni računalnik (slika 10) pomeni enostavno velik računalnik. Med leti 1970 in 1980, ko so bili skoraj vsi računalniki veliki, se je izraz »mainframe« ali centralni računalnik uporabljal za sklicevanje na več različnih računalniških sistemov. Danes večina izmed njih ne obstaja več in centralni ali »mainframe« računalnik se skoraj vedno nanaša na serijo računalnikov »IBM zSeries«. Računalniki »zSeries« so med največjimi na svetu in se uporabljajo za komercialno obdelavo podatkov. Ko govorimo o komercialni obdelavi podatkov, govorimo o aplikacijah, ki temeljijo na podatkovnih bazah. To so aplikacije, katerih glavna naloga je prikazovanje, spreminjanje in brisanje podatkov. Ko poganjamo aplikacije z velikimi podatkovnimi bazami, moramo misliti o:

- celovitosti podatkov (data integrity) – stalna pravilnost podatkov,
- pretoku (throughput) – v danem trenutku želimo opraviti ogromno količino dela,
- odzivu (response) – odgovor želimo takoj, ne v določenem času,
- primeru izpada (disaster recovery) – v primeru izpada želimo biti operativni v čim krajšem možnem času,
- uporabnosti (usability),
- zanesljivosti (reliability),
- reviziji (auditing) – nadzor nad delom uporabnikov,
- varnosti (security).

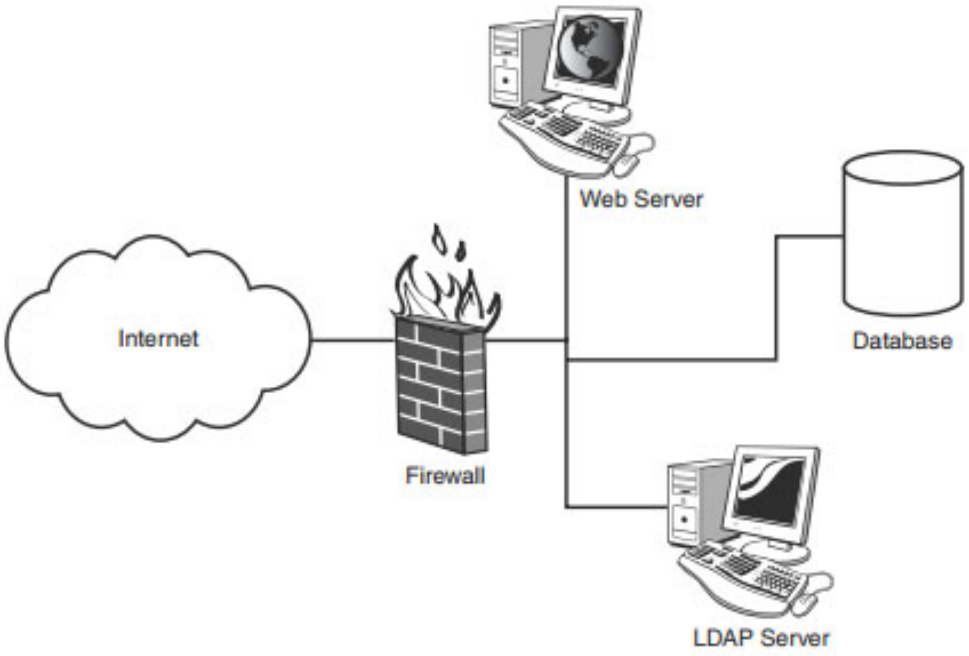
Danes se vsako podjetje, ki želi uporabljati računalnike v komercialne namene, srečuje z zgornjimi težavami. Individualne potrebe, velikost in kompleksnost poslovanja so različni, vendar osnovni problemi ostajajo isti. Dejstvo je, da so veliki računalniki v komercialni obdelavi podatkov zelo dobri. Obstajajo pa tudi stvari, za katere niso primerni, na primer:

- izvajanje dolgih kompleksnih računskih operacij (number crunching),
- grafika in geografski informacijski sistemi.

Veliki računalniki so bili prvotno narejeni za obdelavo velikega števila poslovnih transakcij in se že več kot 50 let nenehno izboljšujejo in spopadajo z izzivi v obdelovanju poslovnih podatkov.
Pri primerjavi velikih računalnikov in računalnikov na drugih platformah vedno igrajo vlogo cena, hitrost, enostavnost. Kaj je bolje, ceneje, enostavneje, je zelo odvisno od naloge, ki jo želimo izvršiti. Na primer študent, ki piše diplomsko nalogo, nima enake potrebe po informacijah kot recimo banka, ki mora obdelati na milijone transakcij vsak dan, predvsem zato, ker mora banka zadostiti tudi vsem varnostnih zahtevam. Veliki računalniki niso za vse naloge. Podjetja se odločijo za centralni računalnik in operacijski sistem »z/OS« [17], ko imajo velike količine podatkov, veliko število transakcijskih obdelav in potrebujejo izredno zanesljiv in varen sistem. Centralni računalnik je dober tudi v primeru, ko imajo podjetja več različnih vrst delovnih obremenitev, ki delujejo najbolje, če se nahajajo na istem računalniku.

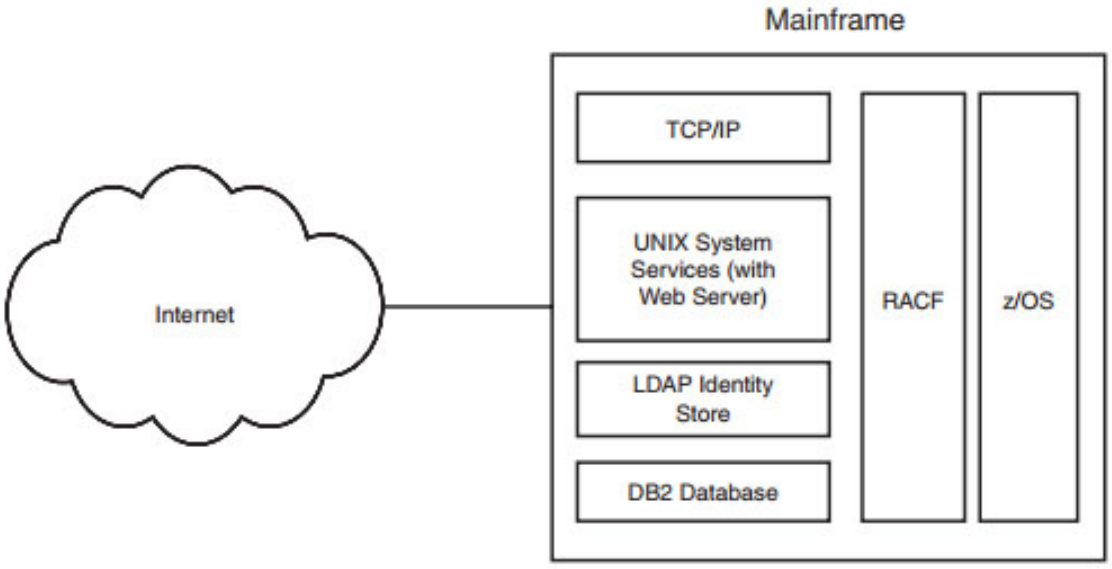
Slika 10: Centralni računalnik IBM (Vir: New IBM zBC12 mainframe gets OpenStack integration. Dostopno na: https://storify.com/ibmevents/new-ibm-zbc12-mainframe-gets-openstack-integration.)

Na kratko opišimo še razliko v delovanju med arhitekturo klient/strežnik in mainframe arhitekturo. V arhitekturi klient/strežnik več strežnikov medsebojno sodeluje, da opravi neko nalogo. Na primer aplikacija uporablja spletni strežnik, strežnik, na katerem je podatkovna baza, in strežnik LDAP (slika 11).



Slika 11: Arhitektura klient – strežnik (Vir: O. Pomerantz, B. Vander Weele, M. Nelson, and T. Hahn (2007): Mainframe Basics for Security Professionals: Getting Started with RACF.)

V mainframe arhitekturi pa centralni računalnik opravlja vse naloge. RACF skrbi za varnost, spletni strežnik, strežnik LDAP, podatkovna baza pa so del enega sistema (slika 12).



Slika 12: Mainframe arhitektura (Vir: O. Pomerantz, B. Vander Weele, M. Nelson, and T. Hahn (2007): Mainframe Basics for Security Professionals: Getting Started with RACF.)

Več o centralnem računalniku, si lahko preberemo v literaturi [2, 3, 13].

4.2 Predstavitev parametrov, ki jih nadziramo

Parametri in komponente, ki jih nadziramo, se delijo v dve skupini. Prva skupina so standardni sistemski viri, ki je jih je nujno treba nadzirati. To so na primer zasedenost centralnih procesnih enot, zasedenost pomnilnika, zasedenost diskovnega polja in drugi. Druga skupina pa so specifični parametri, ki so bili izbrani glede na izkušnje sistemskih skrbnikov in lahko kažejo na morebitne težave na sistemu. To so na primer število transakcij na sekundo, povprečni odzivni čas, performančni indeks in drugi.

Pomembno je omeniti tudi, da za določene parametre, ki jih spremljamo, obstajajo alarmi, ki se vključijo ob kritični porabi določenih virov. Več o tem pa v samem opisu parametrov.

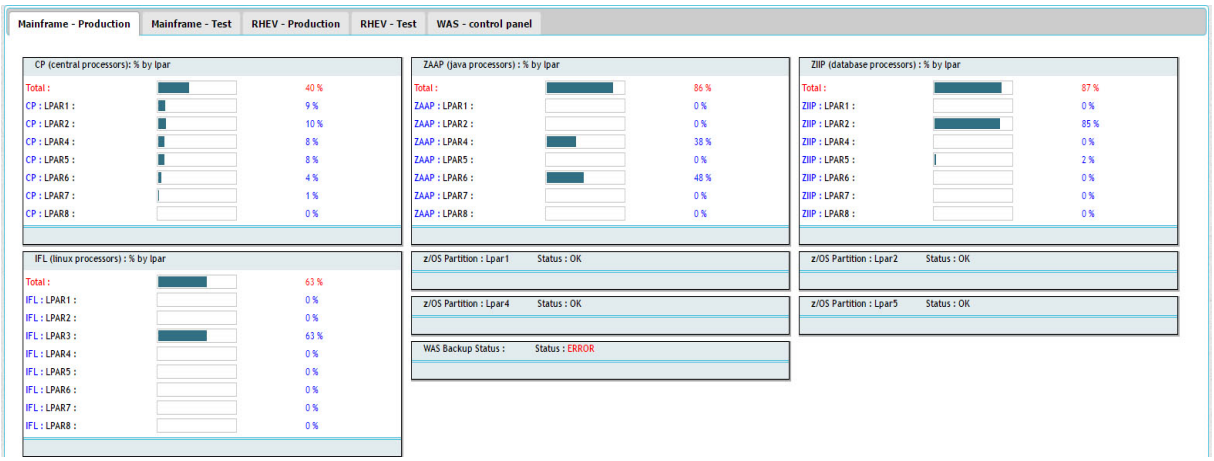
4.2.1 Spremljanje zasedenosti različnih tipov centralnih procesnih enot

Centralni računalnik IBM ima poleg splošnih centralnih procesnih enot tudi več namenskih centralnih procesnih enot. Tipi centralnih procesnih enot, ki so v uporabi v našem primeru, so:

- CP (IBM System z Central Processor) – splošni tip centralne procesne enote, namenjen vsem obdelavam in tipom delovnih obremenitev,
- zIIP (IBM System z Integrated Information Processor) [16] – centralna procesna enota, namenjena baznim obdelavam in razbremenitvi splošnih centralnih procesnih enot,
- zAAP (IBM System z Application Assist Processor) [14] – centralna procesna enota, namenjena obdelavam v Javi in XML-u, za razbremenitev splošnih centralnih procesnih enot,
- IFL (IBM System z Integrated Facility for Linux) [4] – centralna procesna enota, namenjena obdelavam v operacijskih sistemih Linux; tudi te centralne procesne enote so namenjene razbremenitvi splošnih centralnih procesnih enot.

Kot smo ugotovili, so centralne procesne enote namenjene različnim tipom delovnih obremenitev, zato je nujno treba spremljati njihovo zasedenost. Prekomerna zasedenost kateregakoli tipa centralne procesne enote posledično pomeni daljše odzivne čase aplikacij, manjše število obdelanih transakcij na sekundo, več čakalnih vrst za obdelave in posledično nezadovoljne uporabnike, čakalne vrste v uradniških pisarnah in podobno. V skrajnem primeru, ko zasedenost kateregakoli tipa centralne procesne enote doseže 100 %, se začnejo pojavljati bolj obsežne težave. Popolna zasedenost namenskih centralnih procesnih enot pomeni, da del svojih nalog prepustijo splošnim tipom centralnih procesnih enot. Ko tudi

splošne centralne procesne enote dosežejo maksimalno zasedenost, pride do kompleksnejšega izpada sistema. Aplikacije se ustavijo, sporočila se ne obdelujejo, podatkovne baze stojijo, posledično pride do prekinitev v aplikacijskih strežnikih. Potreben je ponovni zagon aplikacijskih strežnikov, podatkovnih baz itd. Aplikacije ne delujejo – prav to je stanje, ki se mu poskušamo izogniti s spremljanjem zasedenosti centralnih procesnih enot in pravočasnim ukrepanjem (slika 13).



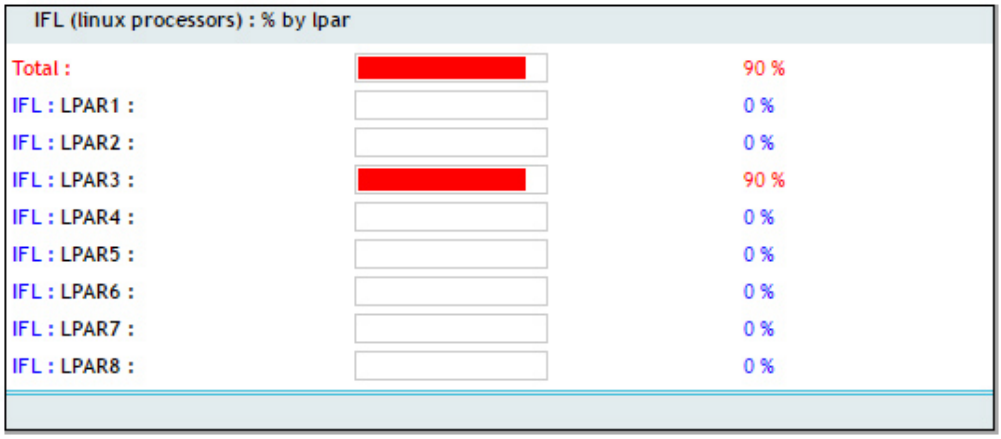
Slika 13: Spremljanje zasedenosti centralnih procesnih enot po posameznih logičnih particijah

V primeru prekomerne zasedenosti centralnih procesnih enot ukrepamo tako, da poskusimo razbremeniti sistem do te mere, da končni uporabniki ne čutijo posledic. Ali smo v tem uspešni, pa je odvisno od tega, ali pravočasno ugotovimo prezasedenost centralnih procesnih enot. V določenih primerih samo spremljanje ni dovolj, saj stalno ne opazujemo spletne aplikacije za nadzor. V takih primerih nam delo olajšajo alarmi.

Alarmi so del aplikacije, ki nam omogočajo zgodnje odkrivanje prezasedenosti centralnih procesnih enot. Možnih alarmov je več:

- povezava s pogovornim sistemom aplikacije Lotus Notes (aplikacija ob alarmu pošlje sporočilo),
- pošiljanje elektronske pošte,
- pošiljanje sporočila SMS preko strežnika sms.gov.si,
- sprememba barv v sistemu za nadzor z namenom hitrejšega opažanja.

Trenutno je v uporabi rešitev s spremembo barve v sistemu za nadzor (slika 14), saj je ta rešitev najbolj primerna za nadzor komponent, katerih vrednosti se hitro spreminjajo in njihova trenutna maksimalna vrednost ni nujno problematična.



Slika 14: Alarm pri zasedenosti centralne procesne enote, ko je zasedenost večja ali enaka 90 %

4.2.2 Spremljanje zasedenosti glavnega pomnilnika

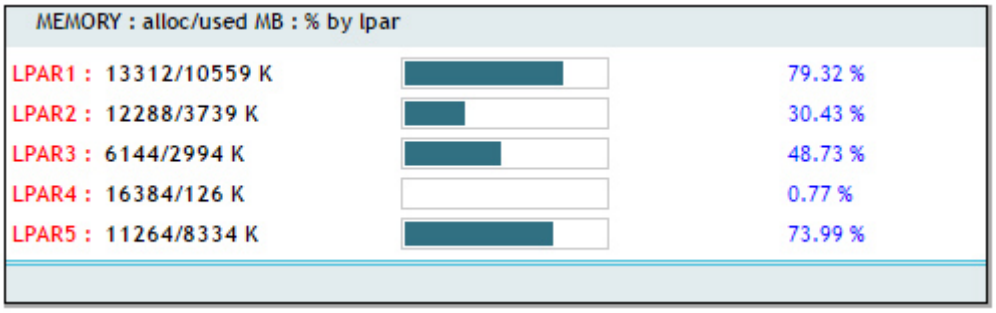
Glavni pomnilnik v računalniku je integrirano vezje, na katerega shranjujemo informacije. Torej se vsak program, kot na primer podatkovna baza, aplikacijski strežnik, spletni strežnik, strežnik LDAP, ob zagonu shrani v del glavnega pomnilnika. Glavni vzrok za to je, da je dostop do glavnega pomnilnika veliko hitrejši kot dostop do trdega diska. Podatki, ki se prenašajo preko teh programov, pa se shranjujejo na trdi disk. Nekateri izmed omenjenih programov rezervirajo določen del glavnega pomnilnika za svoje delovanje. Poglejmo si dva pomembna primera

- Podatkovna baza: podatkovne baze shranjujejo informacije delno v glavni pomnilnik (RAM), delno na trdi disk. Namen tega je, da je dostop do podatkov v glavnem pomnilniku velikokrat hitrejši kot dostop do podatkov na trdem disku.

- Aplikacijski strežnik: večina aplikacijskih strežnikov v programskem jeziku Java ima rezerviran kos glavnega pomnilnika, ki mu pravimo javanska kopica (java heap). Znotraj te kopice se odvija vsa dejavnost aplikacije, ki teče na aplikacijskem strežniku.

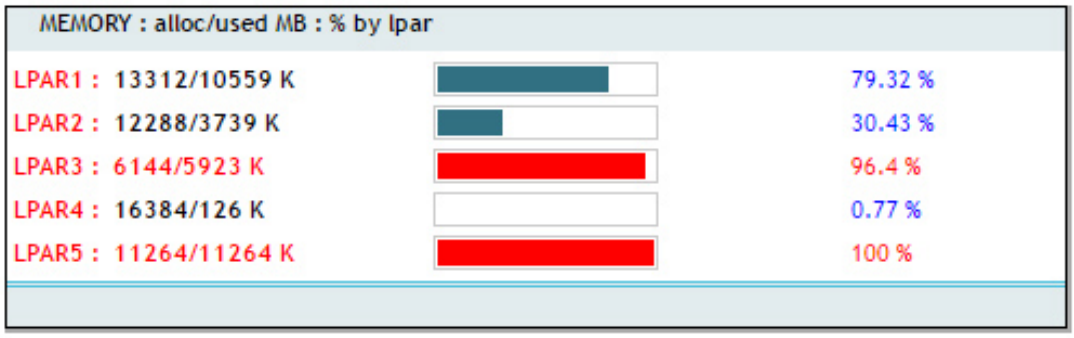
Spremljanje zasedenosti glavnega pomnilnika (slika 15) je v našem primeru pomembno, ker imajo tako podatkovne baze kot tudi aplikacijski strežniki spodnjo in zgornjo mejo rezerviranega glavnega pomnilnika. Zasedenost rezerviranega glavnega pomnilnika se lahko pri množičnih obdelavah podatkov in številnih transakcijah dviga. Ob določenih kritičnih trenutkih lahko skupaj z drugimi dogodki na sistemu preseže maksimalen rezerviran glavni pomnilnik za posamezno logično particijo. V tem primeru pride do težav. Ko glavnega pomnilnika ni več, se začne uporabljati virtualni ali sekundarni pomnilnik (po navadi trdi disk), pri čemer pa je dostop do podatkov veliko počasnejši kot dostop do podatkov v glavnem pomnilniku. Prekomerna zasedenost glavnega pomnilnika posledično pomeni daljše

odzivne čase aplikacij, manjše število obdelanih transakcij na sekundo, več čakalnih vrst za obdelave in posledično nezadovoljne uporabnike, čakalne vrste v uradniških pisarnah in podobno. Ukrepi v takih primerih so nujni, da preprečimo popoln izpad aplikacij.



Slika 15: Zasedenost glavnega pomnilnika po posameznih logičnih particijah

V primeru povečane zasedenosti glavnega pomnilnika ukrepamo enako kot v razdelku 4.2.1. V pomoč so nam alarmi, s pomočjo katerih hitreje odkrivamo težave. Tudi pri spremljanju zasedenosti glavnega pomnilnika so možni enaki tipi alarmov kot pri spremljanju zasedenosti centralnih procesnih enot. V uporabi je samo rešitev s spremembo barve v sistemu za nadzor (slika 16), saj povečanje porabe glavnega pomnilnika ni tako hitro in omogoča dovolj časa za ukrepe, preden je porabljen celoten glavni pomnilnik.

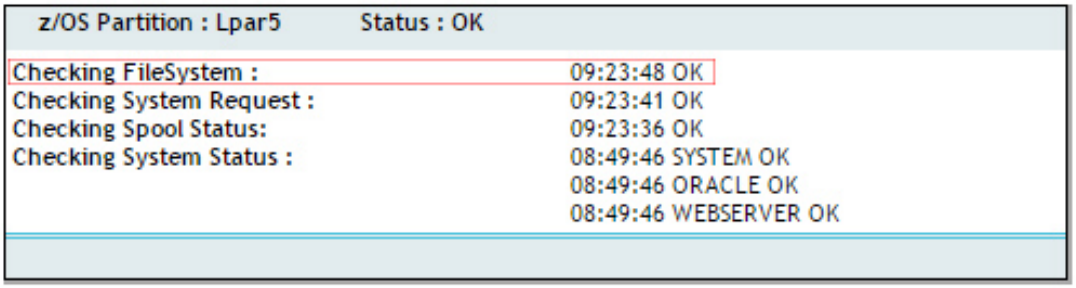


Slika 16: Alarm pri zasedenosti glavnega pomnilnika, ko je zasedenost večja ali enaka 80 %

4.2.3 Spremljanje zasedenosti diskovnih kapacitet

Trdi disk je naprava za shranjevanje podatkov. Je najbolj razširjena oblika sekundarnega pomnilnika. Ob izklopu hrani informacije. Centralni računalnik v svojem ogrodju nima trdih diskov. Nanj je preko optične povezave priklopljeno eno ali več diskovnih polj, na katerih se shranjujejo podatki. Diskovno polje je več fizičnih diskov, predstavljenih kot logična celota, povezanih redundantno, s čimer je zagotovljena večja zmogljivost in zanesljivost.

V našem primeru je spremljanje zasedenosti diskovnih kapacitet zelo pomembno (slika 17), saj vse vrste strežnikov (strežnik LDAP, aplikacijski strežnik, spletni strežnik ...) in podatkovnih baz zasedejo določen prostor na logični particiji. Vsak izmed programov, ki zaseda prostor na trdih diskih, potrebuje tudi dodaten prostor za shranjevanje podatkov, pomembnih dnevnikov, začasnih datotek, v primeru težav pa se nanj zapisujejo tudi podatki, pomembni pri iskanju in odkrivanju napak v delovanju podatkovnih baz, aplikacijskih strežnikov in drugih pomembnih strežnikov, ki tečejo na logičnih particijah.



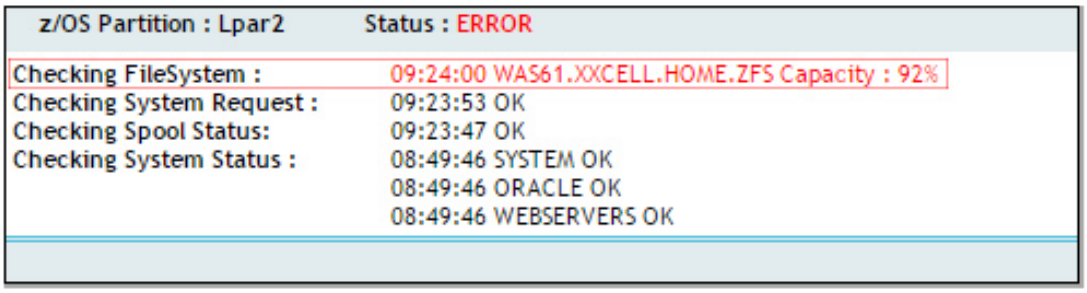
Slika 17: Zasedenost diskovnih kapacitet v primeru normalnega delovanja

V primeru, da se diskovni prostor, ki je dodeljen določeni logični particiji popolnoma zasede, se na sistemu začnejo pojavljati težave. Prihaja do napak v delovanju podatkovnih baz ali celo do popolnega izpada podatkovne baze, saj prostora za zapis podatkov ni in podatkovna baza ne more shranjevati novih podatkov. Tudi pri aplikacijskih in drugih strežnikih se začnejo pojavljati napake, ki lahko vodijo k delnemu ali popolnemu izpadu delovanja aplikacij, to pa je stanje, katerega si ne želi noben sistemski skrbnik.

V primeru, ko se diskovni prostor porabi preko neke določene vrednosti, v našem primeru 90 %, je pomembno, da nas aplikacija opozori na preveliko zasedenost (slika 18). Obvestilo o zasedenosti je nastavljeno na 90 % zato, da imamo še vedno dovolj časa za ukrepanje. Ukrepi v primeru prekomerne zasedenosti diskovnega prostora v neki logični particiji so naslednji:

- zmanjšanje zasedenosti z brisanjem nepotrebnih logov,
- brisanje začasnih datotek,
- premik arhivskih logov,
- premik ali brisanje katerihkoli datotek, ki jih trenutno ne potrebujemo.

V primeru, da tudi katera od zgornjih rešitev ne bi pomagala, pa je treba ustaviti vse strežnike in podatkovne baze in dodati diske logični particiji in nato dodati prostor datotečnem sistemu, ki ga potrebuje. Ta rešitev je »najdražja«, saj se zaradi ponovnega zagona strežnikov in podatkovnih baz izpadu delovanja aplikacij ne moremo izogniti.



Slika 18: Zasedenost diskovnih kapacitet preko 90 %

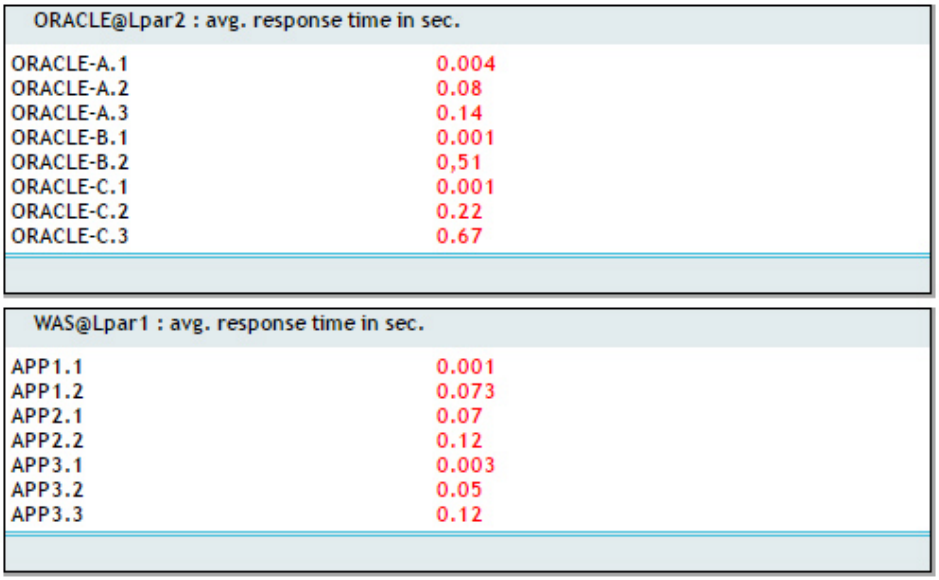
Aplikacija omogoča tudi uporabo alarmov ob prekomerni zasedenosti prostora na enem izmed datotečnih sistemov znotraj logične particije. Možni so enaki tipi alarmov kot pri spremljanju zasedenosti centralnih procesnih enot. V uporabi je samo rešitev s spremembo barve v sistemu za nadzor (slika 18), saj nam ta omogoča dovolj časa za pravočasno ukrepanje.

4.2.4 Spremljanje povprečnega odzivnega časa po posameznih servisnih klasah

Operacijski sistem centralnega računalnika je sestavljen iz množice komponent, ki skrbijo za čim bolj optimizirano delovanje sistema in obdelav, ki tečejo na centralnem računalniku. Ena izmed pomembnih komponent je upravljalnik obremenitev ali WLM (Workload Manager). WLM nadzira dostop do sistemskih virov glede na nastavljene cilje in plane izvajanja.

Na centralnem računalniku se izvaja veliko aplikacij istočasno. Nekatere aplikacije so bolj pomembne kot druge in pri njih je treba poskrbeti, da so odzivni časi v skladu z željami in pričakovanji uporabnika. To dosežemo tako, da nastavimo prioritete in plane izvajanja za določene aplikacije. Sistemski skrbnik razdeli delo med tako imenovane servisne klase (service class). Delo se po servisnih klasah lahko deli glede na ime aplikacije, spletni naslov, virtualni naslovni prostor (AS) in številne druge spremenljivke. Vsaka izmed servisnih klas ima določeno pomembnost (priority), glede na to pomembnost pa se servisni klasi dodeli dostop do sistemskih virov in tudi količina sistemskih virov, ki jih sme zasesti. Obdelava v servisni klasi je časovno omejena. Če se izvajanje ne dokonča v predvidenem času, se obdelava premakne v drugo servisno klaso, definirano za dano obdelavo. V drugi servisni klasi so privilegiji nižji, dostop do sistemskih virov omejen, po navadi pa se čas izvajanja podaljša. Če je treba, se lahko definirajo tudi višje servisne klase z manj rezerviranimi sistemskimi viri in zelo dolgim časom izvajanja. Te so po navadi namenjene paketnim obdelavam (batch jobs), ki so dolgotrajne in jim je treba omejiti dostop do sistemskih virov, saj lahko povzročijo dolgotrajno pretirano porabo sistemski virov.

V našem primeru spremljamo povprečne odzivne čase aplikacij in podatkovnih baz Oracle (slika 19). Odzivni čas je čas od trenutka, ko neka zahteva vstopi v sistem, do trenutka, ko je ta zahteva obdelana. S spremljanjem povprečnih odzivnih časov po servisnih klasah ugotavljamo, kako hitro se odzivajo aplikacije in podatkovne baze Oracle, ali se zahteve obdelujejo v zahtevanem času in ali sistem deluje optimalno. V primeru, da se odzivni časi višajo in vse več obdelav prehaja v drugo ali celo tretjo servisno klaso, je to kazalnik, da se na sistemu pojavljajo težave. V tem primeru se odzovemo tako, da začnemo spremljati tudi druge komponente sistema, omrežne povezave, preverjamo same aplikacije in glede na opažene težave ustrezno ukrepamo.



ORACLE@Lpar2 : avg. response time in sec.

ORACLE-A.1	0.004
ORACLE-A.2	0.08
ORACLE-A.3	0.14
ORACLE-B.1	0.001
ORACLE-B.2	0,51
ORACLE-C.1	0.001
ORACLE-C.2	0.22
ORACLE-C.3	0.67

WAS@Lpar1 : avg. response time in sec.

APP1.1	0.001
APP1.2	0.073
APP2.1	0.07
APP2.2	0.12
APP3.1	0.003
APP3.2	0.05
APP3.3	0.12

Slika 19: Povprečni odzivni čas po posameznih servisnih klasah

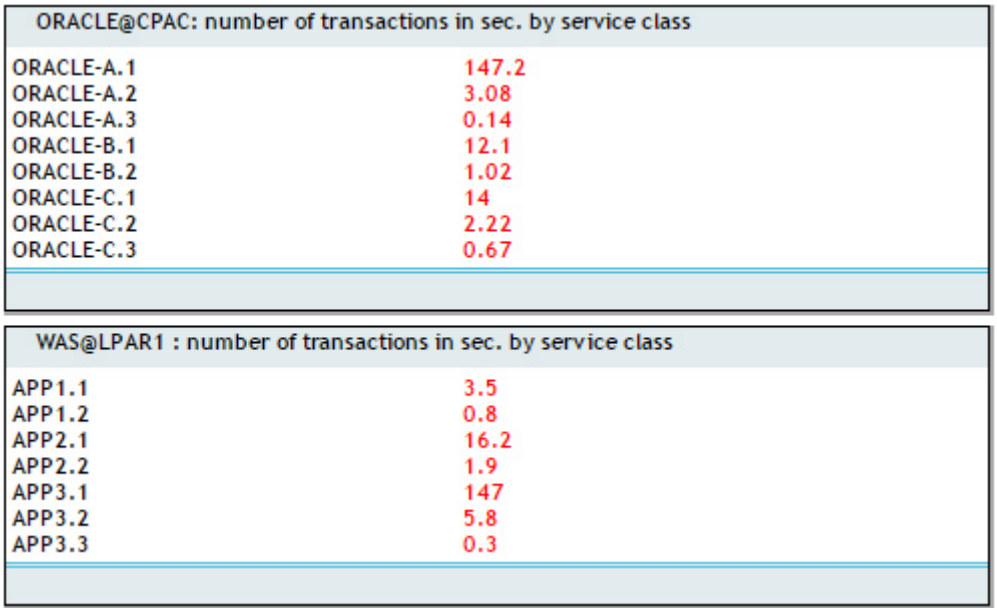
4.2.5 Spremljanje števila transakcij po posameznih servisnih klasah

Kot smo že omenili, ima operacijski sistem centralnega računalnika tako imenovani upravljalnik obremenitev (workload manager), ki določa, katere obdelave se bodo izvršile prej, katere kasneje in katere bodo dobile več dostopa do sistemskih virov. Plane izvajanja določimo za dane aplikacije, podatkovne baze, aplikacijske in druge strežnike s pomočjo servisnih klas, katerim dodelimo delo.

Kot je pomembno spremljati osnovne sistemske vire, kot so zasedenost procesorja, spomina, diskovnih kapacitet, je pomembno spremljati tudi, kako se aplikacije in podatkovne baze odzivajo (povprečni odzivni čas), kar smo že omenili. Poleg vsega tega pa nas zanima, koliko

dela naše aplikacije izvedejo na sekundo. To ugotovimo s spremljanjem števila transakcij, ki jih neka servisna klasa obdela v sekundi. Aplikacije in podatkovne baze so razdeljene po servisnih klasah, torej lahko s spremljanjem števila transakcij na sekundo po posamezni servisni klasi ugotavljamo, koliko transakcij je aplikacija ali podatkovna baza izvedla na sekundo (slika 20).

Po določenem času spremljanja števila transakcij na sekundo za določene aplikacije in podatkovne baze iz izkušenj vemo, kateri rezultati pomenijo, da sistem normalno deluje in da aplikacije ter podatkovne baze dosegajo optimalne rezultate pri obdelavah. V primeru, da teh rezultatov ne dosegamo, je možno, da se na sistemu dogajajo težave. S spremljanjem drugih komponent centralnega računalnika potem ugotavljamo vrsto težav in ukrepe za izboljšanje delovanja.



ORACLE@CPAC: number of transactions in sec. by service class

ORACLE-A.1	147.2
ORACLE-A.2	3.08
ORACLE-A.3	0.14
ORACLE-B.1	12.1
ORACLE-B.2	1.02
ORACLE-C.1	14
ORACLE-C.2	2.22
ORACLE-C.3	0.67

WAS@LPAR1 : number of transactions in sec. by service class

APP1.1	3.5
APP1.2	0.8
APP2.1	16.2
APP2.2	1.9
APP3.1	147
APP3.2	5.8
APP3.3	0.3

Slika 20: Število izvedenih transakcij na sekundo po posameznih servisnih klasah za podatkovne baze Oracle in aplikacije, ki tečejo na aplikacijskih strežnikih IBM WebSphere

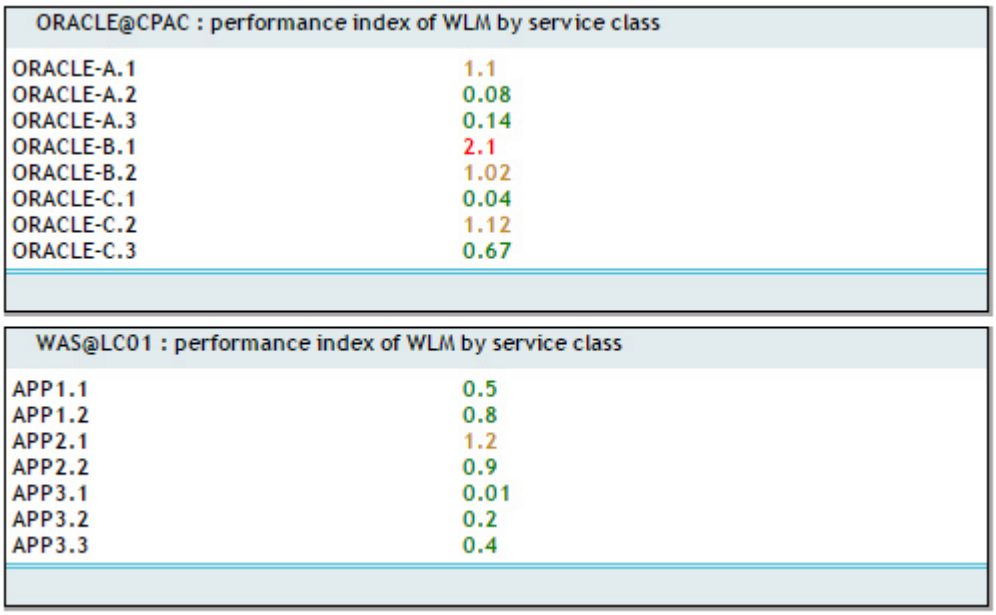
4.2.6 Spremljanje performančnega indeksa WLM-ja po posameznih servisnih klasah

Izmed vseh dobrih lastnosti upravljalnika obremenitev je treba omeniti še eno. Poleg planov izvajanja za določeno servisno klaso lahko določimo tudi cilj, ki ga želimo doseči. Spremenljivka, ki nam opisuje ta cilj, se imenuje performančni indeks WLM-ja.

Performančni indeks za posamezno servisno klaso je rezultat izračuna določenih spremenljivk, ki nam pove, ali je bil nastavljeni cilj dosežen, presežen ali pa cilja nismo dosegli. Možni rezultati so:

- performančni indeks = 1, cilj je dosežen,
- performančni indeks < 1, cilj je presežen,
- performančni indeks > 1, cilja nismo dosegli.

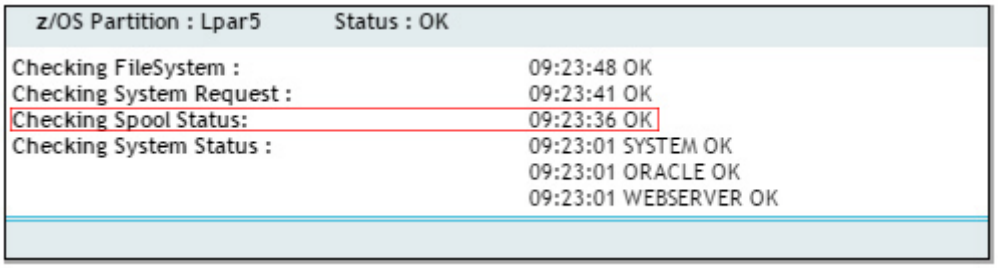
Spremljanje performančnega indeksa (slika 21) nam omogoča spremljanje nastavljenih ciljev. Če ciljev nismo dosegli, so potrebni ukrepi na sistemu in spremembe nastavitev WLM-ja in servisnih klas, ki bodo izboljšali delovanje in dosegali zastavljene cilje.



Slika 21: Spremljanje performančnega indeksa po posamezni servisni klasi za podatkovne baze Oracle in aplikacije, ki tečejo na aplikacijskih strežnikih IBM WebSphere

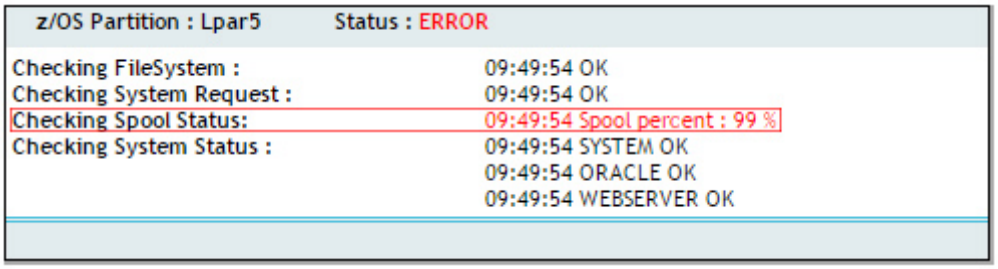
4.2.7 Spremljanje delovanja komponente SPOOL po posameznih logičnih particijah

SPOOL (Simultaneous Peripheral Operations On-Line) je repozitorij vseh vhodnih opravil in večine izhodnih opravil (sysout) na centralnem računalniku. V primeru, ko je prostor namenjen za SPOOL, izkoriščen, se na sistemu začnejo pojavljati performančne težave. Ker v določenih primerih lahko pride celo do izpada sistema, je spremljanje zasedenosti prostora, rezerviranega za SPOOL, zelo priporočljivo (slika 22).



Slika 22: Spremljanje zasedenosti prostora, rezerviranega za SPOOL

Da do teh težav ne pride, nam spletna aplikacija ob zasedenosti prostora, namenjenega za SPOOL, nad 90 %, javi alarm (slika 23).

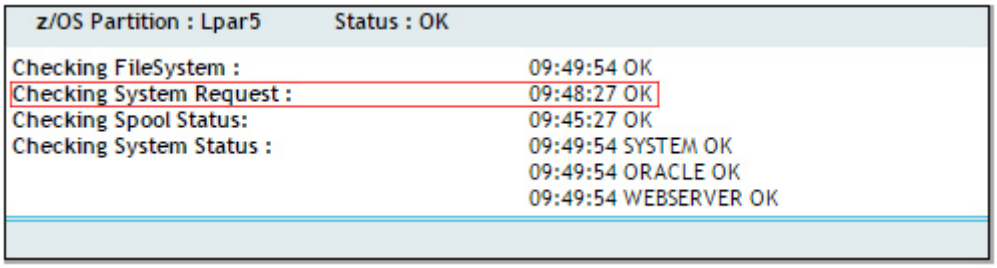


Slika 23: Alarm pri zasedenosti prostora, namenjenega za SPOOL, nad 90 %

V primeru, ko se zasedenost prostora, namenjenega za SPOOL, bliža 100 %, ukrepamo tako, da ob primernem času, ko zunanji uporabnik ne bo čutil posledic, povečamo diskovni prostor, namenjen za SPOOL.

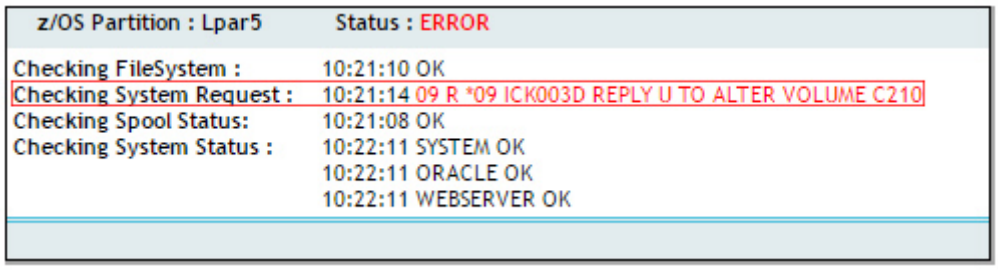
4.2.8 Spremljanje sistemskih zahtev po posameznih logičnih particijah

Sistemske zahteve (system request) na centralnem računalniku so pomembne, saj preko njih operacijski sistem centralnega računalnika pove operaterju ali sistemskemu skrbniku, katere akcije se dogajajo na sistemu in ali je zanje potrebna potrditev. Če je potrebna potrditev, sistemski skrbnik v terminalu odgovori na sistemsko zahtevo. Centralni računalnik ima lahko večje število logičnih particij, vsaka logična particija pa ima svoj operacijski sistem. Pri večjem številu operacijskih sistemov je težko nadzirati vse sistemske zahteve. V tem primeru si zadevo olajšamo s spletno aplikacijo, ki namesto nas spremlja sistemske zahteve (slika 24).



Slika 24: Spremljanje sistemskih zahtev na centralnem računalniku

Če operacijski sistem logične particije centralnega računalnika poda neko sistemsko zahtevo, to vidimo v spletni aplikaciji (slika 25).



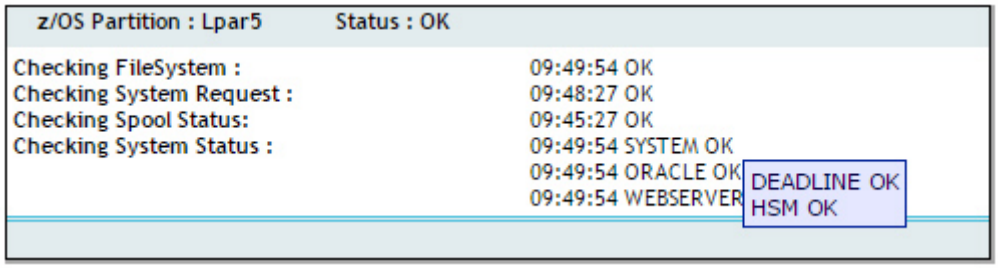
Slika 25: Primer sistemske zahteve na operacijskem sistemu logične particije centralnega računalnika

V takem primeru se operater ali sistemski skrbnik ustrezno odzove in v terminalu odgovori na dano sistemsko zahtevo.

4.2.9 Spremljanje komponent AS po posameznih logičnih particijah

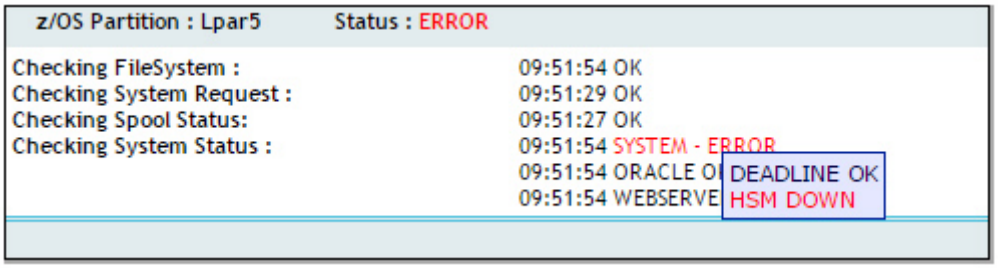
Zadnja in hkrati zelo pomembna komponenta, ki jo spremljamo, se imenuje AS. Address space je virtualni naslovni prostor, ki ga operacijski sistem centralnega računalnika dodeli neki aplikaciji. Virtualni naslovni prostor na operacijskem sistemu z/OS pomeni v primerjavi z operacijskim sistemom Windows ali Linux proces neke aplikacije.

Spletna aplikacija za nadzor nam omogoča spremljanje vseh pomembnih naslovnih prostorov v operacijskem sistemu neke logične particije centralnega računalnika (slika 26). V našem primeru so naslovni prostori razdeljeni v skupine (System, Oracle, Webserver). Ob premiku kurzorja na eno izmed skupin se odpre novo okno s stanjem naslovnih prostorov (slika 26).



Slika 26: Spremljanje naslovnih prostorov operacijskega sistema logične particije centralnega računalnika

Vsak naslovni prostor predstavlja neko aplikacijo, aplikacijski strežnik, podatkovno bazo, strežnik LDAP, spletni strežnik, sistemske programske komponente ipd. S pomočjo spremljanja naslovnih prostorov enostavno ugotavljamo, ali vse aplikacije, aplikacijski in drugi strežniki, podatkovne baze dejansko delujejo. Če kateri izmed naslovnih prostorov ne deluje, se v spletni aplikaciji prižge alarm (slika 27). Sistemski skrbnik ukrepa tako, da ugotovi, zakaj naslovni prostor ne deluje, in ga po potrebi ponovno zažene.

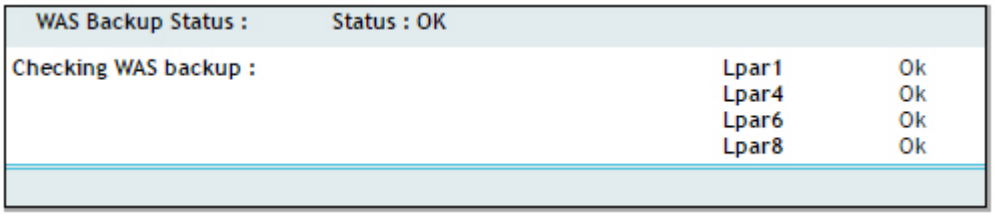


Slika 27: Alarm ob nedelovanju enega izmed naslovnih prostorov

4.2.10 Spremljanje izvajanja varnostnih kopij aplikacijskih strežnikov

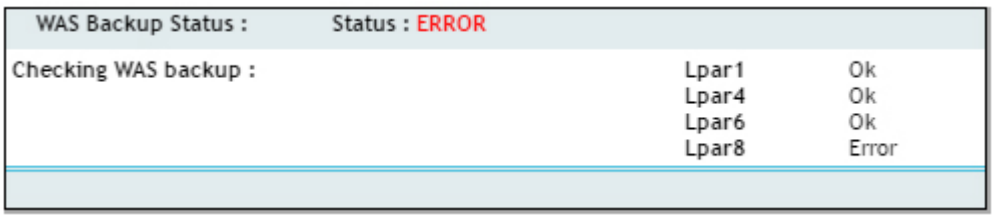
Delovanje aplikacijskih strežnikov je zelo pomembno, saj na njih tečejo vse pomembne aplikacije. Včasih pa se zgodi, da pride do izpada delovanja aplikacijskih strežnikov in s tem aplikacij tudi ob vseh ukrepih in spremljanju delovanja sistemov. V primeru, ko pride do izpada aplikacijskih strežnikov zaradi okvare podatkov, okvare trdih diskov ali česa podobnega, je nujno treba imeti varnostne kopije aplikacijskih strežnikov.

Varnostne kopije se izvedejo enkrat na dan za vsak aplikacijski strežnik na vsaki logični particiji centralnega računalnika. V primeru velikega števila aplikacijskih strežnikov je dnevno preverjanje, ali se je varnostna kopija ustvarila, zelo zamudno. Spletna aplikacija pa nam omogoča spremljanje izvajanja varnostnih kopij za vsak aplikacijski strežnik (slika 28). Spletna aplikacija preveri, ali je varnostna kopija izvedena in ali ni starejša od enega dneva.



Slika 28: Spremljanje izvajanja varnostnih kopij aplikacijskih strežnikov IBM WebSphere

V primeru, da varnostna kopija ni izvedena ali je starejša od enega dneva, spletna aplikacija javi napako (slika 29).



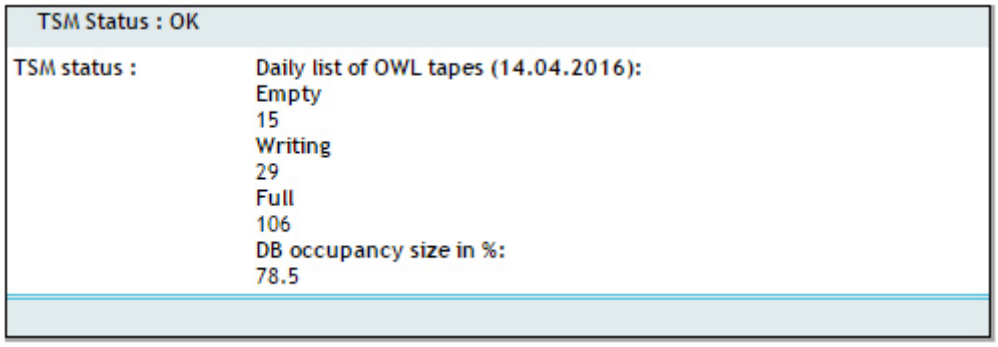
Slika 29: Spremljanje izvajanja varnostnih kopij aplikacijskih strežnikov IBM WebSphere v primeru, ko varnostna kopija ni bila izvedena ali je starejša od enega dneva

4.2.11 Spremljanje delovanja strežnika TSM

IBM Tivoli Storage Manager (IBM Spectrum Protect) je platforma za varovanje podatkov ali, bolj enostavno, sistem za varnostno kopiranje podatkov. Omogoča nam zanesljivo varnostno kopiranje in obnovo podatkov. Administracija, statistika, pregled in obnova podatkov potekajo iz centralnega portala, s katerim nadzorujemo in administriramo celotno platformo.

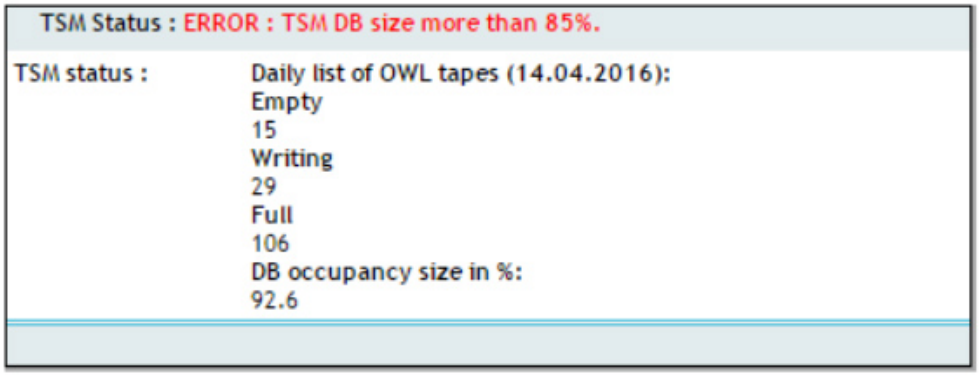
TSM uporablja relacijsko bazo in obnovitveni dnevnik (recovery log) za obnovitev, konfiguracijo, statistiko in za objektne metapodatke. Aktualni uporabniški podatki se upravljajo preko kaskadne hierarhije medijev za shranjevanje (primary storage pools), ki so predstavljeni kot diski, diskovna polja, trakovi, optični mediji in drugi mediji za shranjevanje.

Zaradi možnosti okvare diskovnih polj, trdih diskov ali izgube podatkov na strežnikih iz katerihkoli razlogov je pomembno spremljati varnostno kopiranje posameznih strežnikov. Spremljanje varnostnega kopiranja posameznih strežnikov poteka na samih strežnikih. Seznami se pošiljajo preko elektronske pošte sistemskim skrbnikom, ki so odgovorni za določen strežnik. Vseeno pa je pomembno spremljati tudi delovanje sistema za varnostno kopiranje. Spletna aplikacija nam omogoča dnevno spremljanje števila prostih, zasedenih medijev in medijev, na katere se pravkar zapisujejo podatki, ter zasedenost relacijske podatkovne baze (slika 30).



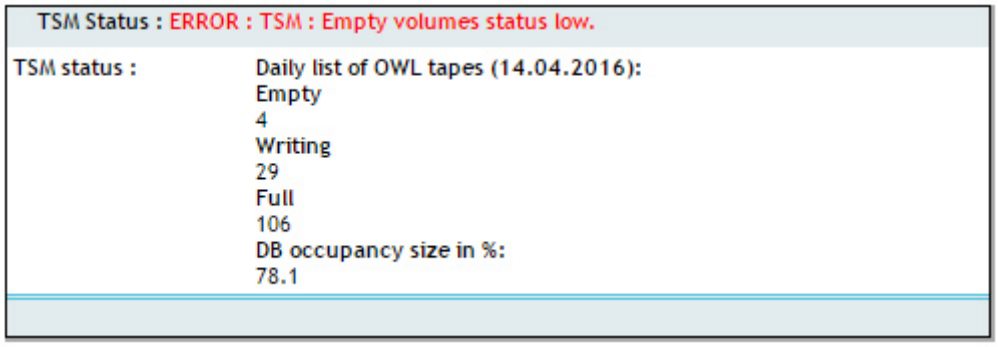
Slika 30: Spremljanje števila medijev za shranjevanje in zasedenost relacijske baze

Če je zasedenost relacijske baze več kot 85 %, se prižge alarm (slika 31) in »TSM Status« na vrhu okna prikazuje dano napako. V takem primeru je časa za ukrepanje še nekaj, zato je alarm nastavljen precej nizko. V primernem trenutku, ko uporabniki tega ne občutijo, se sistem za varnostno kopiranje izklopi in relacijski bazi se dodeli več prostora.



Slika 31: Alarm pri prezasedenosti relacijske podatkovne baze

V primeru, ko začne primanjkovati medijev za shranjevanje, nam spletna aplikacija javi alarm (slika 32) in »TSM Status« na vrhu okna prikazuje vrsto napake. V takem primeru je časa za ukrepanje še nekaj, saj so mediji za shranjevanje po navadi precej veliki. V primernem trenutku, ko uporabniki tega ne občutijo, se sistem za varnostno kopiranje izklopi in sistemu se dodajo prosti mediji.



Slika 32: Alarm pri pomanjkanju medijev za varnostno kopiranje

5 Nadzor delovanja strežnikov z operacijskim sistemom Linux

V tem poglavju so predstavljeni operacijski sistem Linux [8] in parametri, ki jih na operacijskem sistemu Linux spremljamo, njihov pomen in kako ukrepati v primeru, da njihove vrednosti niso v mejah optimalnega delovanja.

5.1 Predstavitev operacijskega sistema Linux

Linux je najbolj znan in najbolj uporabljen odprtokodni operacijski sistem na svetu. Je programska oprema, ki deluje pod vso drugo programsko opremo na računalniku, prejema zahteve od vse druge programske opreme in jih posreduje strojni opremi računalnika. Operacijski sistem Linux je naredil študent Univerze v Helsinkih Linus Torvalds leta 1991. Želel je narediti odprtokodni operacijski sistem po zgledu operacijskega sistema Minix in to mu je tudi uspelo.

V mnogih pogledih je Linux podoben drugim operacijskim sistemom, ki jih vsi uporabljamo (Windows, OS X). Ima grafični vmesnik in programsko opremo, ki smo jo vajeni tudi na drugih operacijskih sistemih, a hkrati je v več pomembnih pogledih različen od drugih operacijskih sistemov. Prva in mogoče najpomembnejša razlika je, da je Linux odprtokodni operacijski sistem, kar pomeni, da je njegova izvorna koda brezplačna in na voljo javnosti v uporabo ali dopolnitve. Drugačen je tudi s tem, da obstaja veliko različic operacijskega sistema Linux, ki jim pravimo distribucije. Te vključujejo veliko možnosti programske opreme, kar pomeni, da je Linux izjemno prilagodljiv. Njegovi uporabniki lahko izberejo ključne komponente, ki jih želijo v svojem operacijskem sistemu (grafika, uporabniški vmesnik in druge komponente).

Operacijski sistem Linux je namenjen vsem, ki potrebujejo varen in zmogljiv operacijski sistem. Verjetno smo že vsi kdaj uporabljali Linux, pa če smo to vedeli ali ne. Odvisno od raziskav, toda med eno in dvema tretjinama vseh spletnih strani poganja operacijski sistem Linux. Podjetja se odločajo za operacijske sisteme Linux zaradi varnosti in ker nudijo poleg komercialne podpore podjetij, kot so Canonical, SuSe, Red Hat in druga, tudi podporo velike skupnosti uporabnikov.

Operacijski sistem Linux poganja veliko naprav, kot so mobilni telefoni, kamere, digitalne naprave za shranjevanje, pametne ure in celo nekatere znamke avtomobilov ga uporabljajo za poganjanje sistemov za nadzor in zabavo.

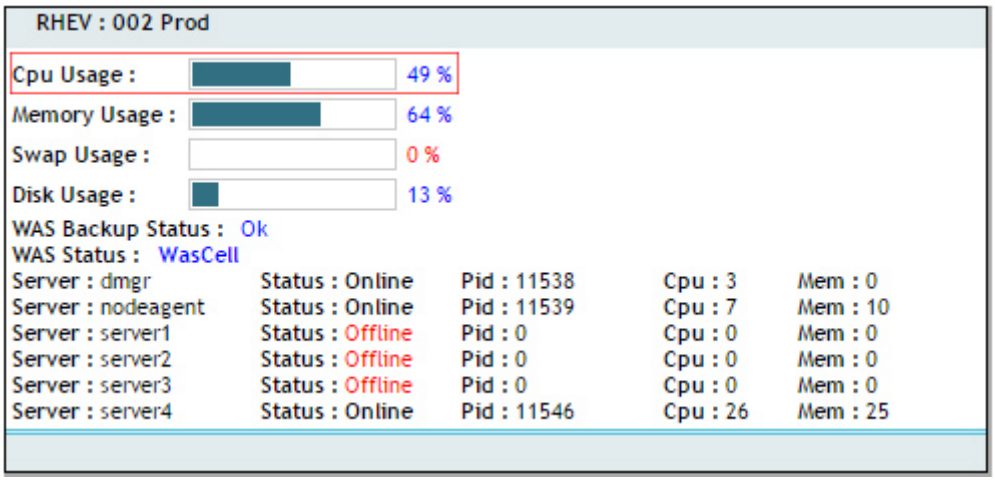
5.2 Predstavitev parametrov, ki jih nadziramo

Parametri in komponente, ki jih nadziramo, se delijo v dve skupini. Prva skupina so standardni sistemski viri, ki je jih je nujno treba nadzirati. To so zasedenost centralnih procesnih enot, zasedenost pomnilnika, zasedenost diskovnega polja, zasedenost izmenjevalnih datotek, izvajanje varnostnih kopij operacijskega sistema in aplikacijskih strežnikov. Druga skupina so komponente, ki so bile izbrane glede na trenutne potrebe in so pomembne za optimalno delovanje aplikacij. To je spremljanje delovanja aplikacijskih strežnikov tipa IBM WebSphere in IBM Lotus Domino.

Pomembno je omeniti tudi, da za določene parametre, ki jih spremljamo, obstajajo alarmi, ki se vključijo ob kritični porabi določenih virov. Več o tem pa v opisu parametrov.

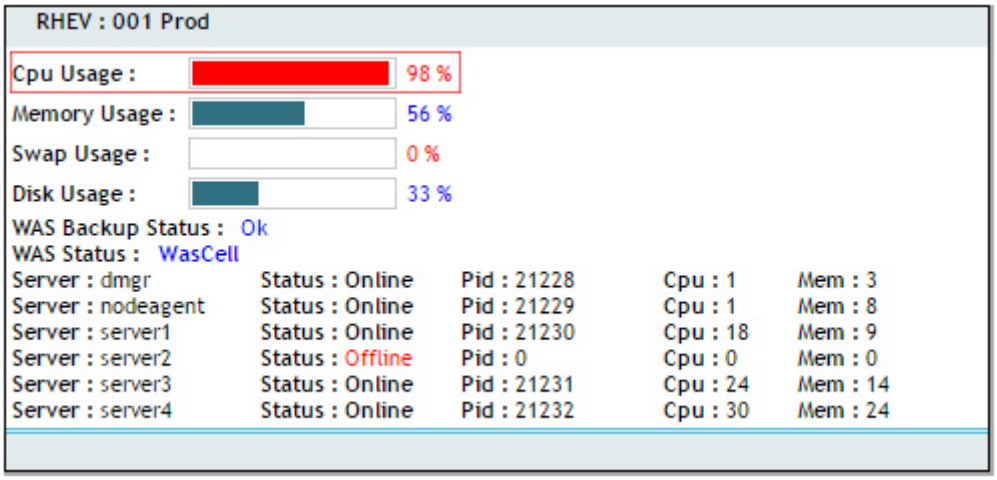
5.2.1 Spremljanje zasedenosti centralnih procesnih enot

Centralne procesne enote strežnikov, na katerih deluje operacijski sistem Linux, so namenjene vsem tipom delovnih obremenitev in nimajo posameznih tipov namensko določenih procesorjev, kot na primer centralni računalnik. Vendar pa prekomerna zasedenost centralnih procesnih enot pomeni približno enako težavo kot na centralnem računalniku. V primeru, ko se zasedenost centralnih procesnih enot približa 100 %, se začnejo pojavljati težave na sistemu. Te težave pomenijo daljše odzivne čase aplikacij, manjše število obdelanih transakcij, čakalne vrste za obdelave in na splošno nezadovoljne končne uporabnike. V primeru, ko centralne procesne enote dosežejo maksimalno zasedenost, ki traja dalj časa, pa pride do popolnega izpada delovanja. V tem primeru ukrepamo tako, da ugotovimo vzrok napake in ga odpravimo. Najverjetneje pa je potem potreben tudi ponovni zagon aplikacijskih strežnikov, kar dodatno podaljša čas izpada aplikacij. V tem času aplikacije ne delujejo, prav to pa je stanje, kateremu se poskušamo izogniti s spremljanjem zasedenosti centralnih procesnih enot in pravočasnim ukrepanjem (slika 33).



Slika 33: Spremljanje zasedenosti centralne procesne enote na strežnikih z operacijskim sistemom Linux

Prav tako kot pri spremljanju zasedenosti centralnih procesnih enot centralnega računalnika imamo tudi tu na voljo alarme, ki nam olajšajo spremljanje zasedenosti centralnih procesnih enot in omogočajo pravočasno ugotavljanje težav. Na voljo so enaki tipi alarmov kot v razdelku 4.2.1, prav tako je v uporabi enak alarm (slika 34).

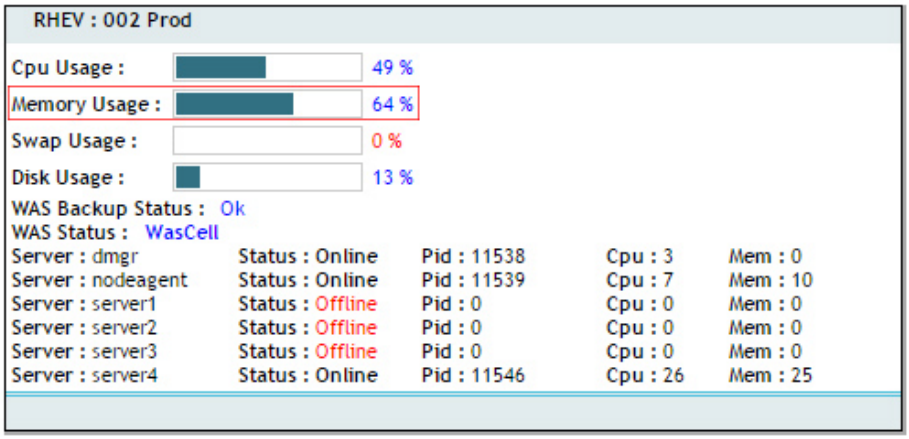


Slika 34: Alarm pri zasedenosti centralnih procesnih enot, ko je zasedenost večja od 90 %

5.2.2 Spremljanje zasedenosti glavnega pomnilnika

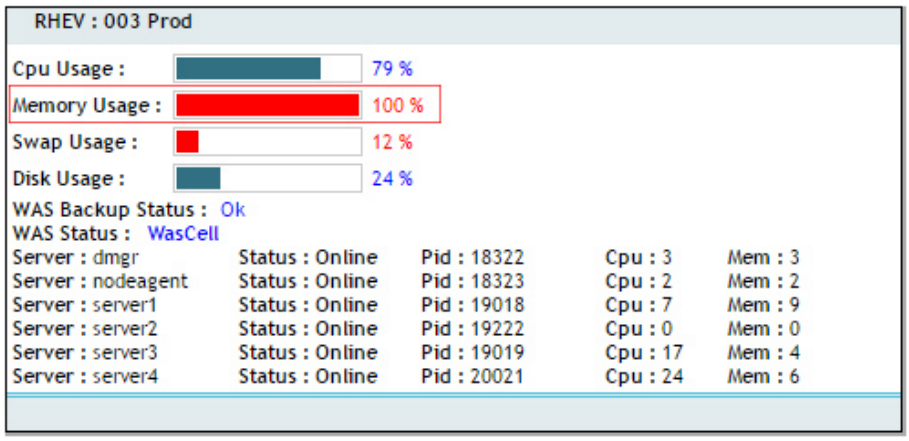
Kot smo že omenili pri centralnem računalniku, je spremljanje zasedenosti glavnega pomnilnika (slika 35) pomembno zato, ker večina aplikacij deluje znotraj glavnega pomnilnika in tudi rezervirajo del glavnega pomnilnika za svoje obdelave.
V določenih kritičnih trenutkih lahko poraba aplikacij skupaj z drugimi dogodki na sistemu povzroči pomanjkanje glavnega pomnilnika. V tem primeru se vsa dejavnost prestavi v sekundarni ali virtualni pomnilnik (po navadi trdi disk), kjer stvari začnejo delovati precej

počasneje in na sistemu se pričnejo pojavljati enake težave, kot smo jih že opisali v razdelku 4.2.2.



Slika 35: Spremljanje zasedenosti glavnega pomnilnika na strežnikih z operacijskim sistemom Linux

V primeru povečane zasedenosti pomnilnika ukrepamo enako, kot smo že omenili v razdelku 4.2.2 Na voljo so enaki tipi alarmov in v uporabi je enak tip alarma kot v razdelku 4.2.2 (slika 36).



Slika 36: Alarm pri zasedenosti glavnega pomnilnika, ko je pomnilnik zaseden več kot 80 %

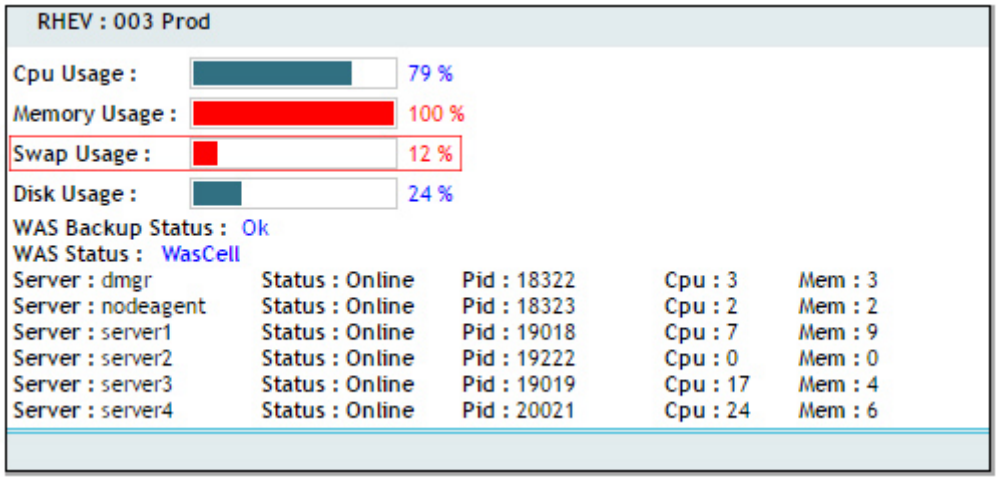
5.2.3 Spremljanje zasedenosti izmenjevalnega prostora (swap space)

Operacijski sistem Linux deli primarni pomnilnik na enako veliko kose, imenovane strani. »Swapping« je proces izmenjevanja strani, ko je stran iz primarnega pomnilnika prepisana v sekundarni virtualni pomnilnik (po navadi trdi disk), imenovan izmenjevalni prostor (swap space), z namenom, da prepisana stran sprosti prostor v glavnem primarnem pomnilniku.

Izmenjevanje strani je nujno iz dveh pomembnih razlogov:

1. v primeru, ko sistem potrebuje več glavnega pomnilnika, kot ga je na voljo, se manj uporabljene strani v pomnilniku prepišejo v izmenjevalni prostor in sprostijo del glavnega pomnilnika;
2. nekateri programi lahko ob zagonu zasedejo veliko število strani v glavnem pomnilniku, kasneje lahko nepotrebne strani izmenjamo in sprostimo del glavnega pomnilnika.

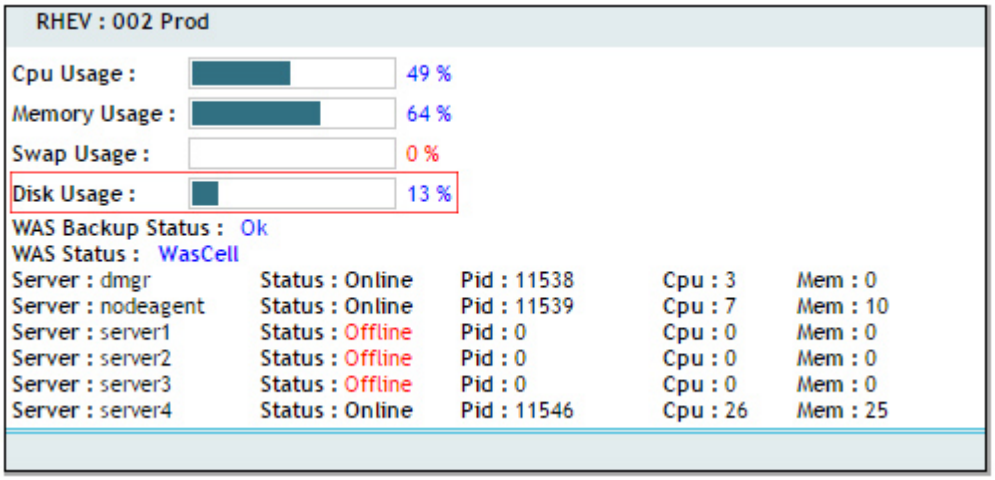
Toda izmenjevanje ima tudi slabo stran. V primerjavi s primarnim pomnilnikom so trdi diski zelo počasni. Več ko je izmenjevanja, bolj počasi deluje sistem. To je glavni razlog, zakaj spremljamo zasedenost izmenjevalnega prostora (slika 37). Že v primeru, ko je zasedenost večja od 0, lahko slutimo, da se bodo na sistemu začele pojavljati težave, saj iz tega sklepamo, da nam primanjkuje glavnega pomnilnika. Posledice primanjkovanja glavnega pomnilnika in ukrepe za izboljšanje stanja pa smo opisali v razdelku 5.2.2.



Slika 37: Spremljanje zasedenosti prostora za izmenjevanje (swap space)

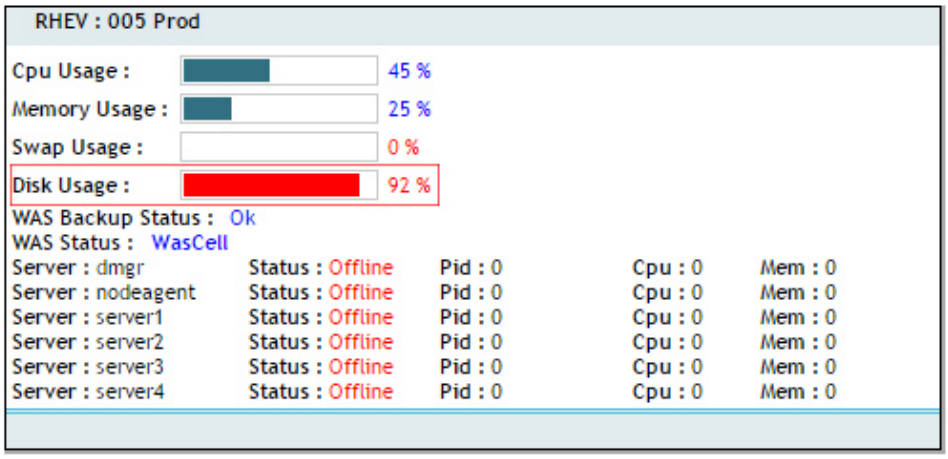
5.2.4 Spremljanje zasedenosti diskovnih kapacitet

Spremljanje zasedenosti diskovnih kapacitet (slika 38) je zelo pomembno, vendar se ne razlikuje veliko od že opisanega v razdelku 4.2.3.



Slika 38: Spremljanje zasedenosti diskovnih kapacitet na operacijskem sistemu Linux

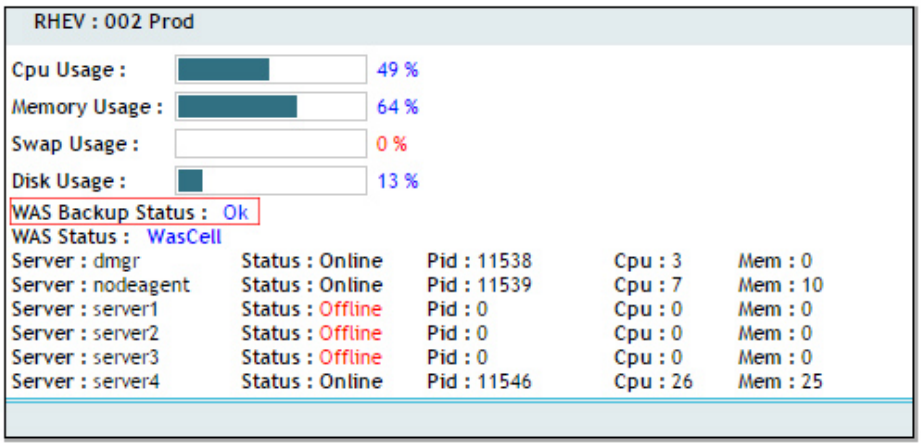
V primeru povečane zasedenosti diskovnih kapacitet ukrepamo enako, kot smo to opisali v razdelku 4.2.3. Poglavitno je, da z ukrepi poskusimo pridobiti dodaten prostor na trdih diskih. Ali bomo uspešni, je odvisno od tega, ali pravočasno ugotovimo povečanje zasedenosti diskovnih kapacitet. V določenih primerih samo spremljanje ni dovolj, saj ne opazujemo stalno spletnih aplikacij. V takih primerih nam delo olajšajo alarmi. Na voljo so enaki tipi alarmov, kot smo jih opisali v razdelku 4.2.3. V uporabi je enak tip alarma, kot smo ga opisali v razdelku 4.2.3 (slika 39).



Slika 39: Alarm pri zasedenosti diskovnih kapacitet preko 90 %

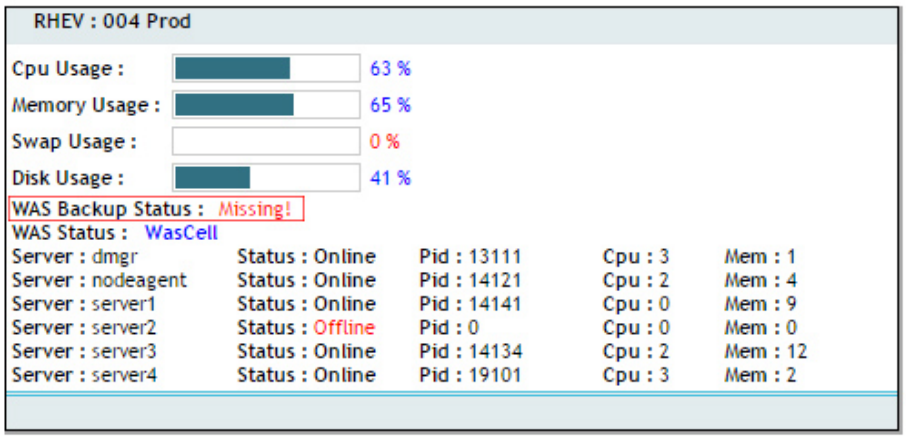
5.2.5 Spremljanje izvajanja varnostnih kopij operacijskega sistema in aplikacijskih strežnikov

Izvajanje varnostnih kopij operacijskega sistema, podatkov in aplikacijskih strežnikov je zelo pomembno, saj v primeru okvare diskov ali podatkov nujno potrebujemo zadnjo verzijo varnostne kopije, s katero obnovimo sistem. Varnostne kopije se izvedejo enkrat na dan. Kopije sistema in podatkov se preverjajo preko elektronske pošte. V primeru neuspele varnostne kopije se pošlje elektronska pošta z navedbo napake. V primeru velikega števila aplikacijskih strežnikov je dnevno preverjanje, ali se je varnostna kopija ustvarila, zelo zamudno. Spletna aplikacija nam omogoča spremljanje izvajanja varnostnih kopij za vsak aplikacijski strežnik (slika 40). Spletna aplikacija preveri, ali je varnostna kopija izvedena in ali ni starejša od enega dneva.



Slika 40: Spremljanje izvajanja varnostne kopije aplikacijskega strežnika

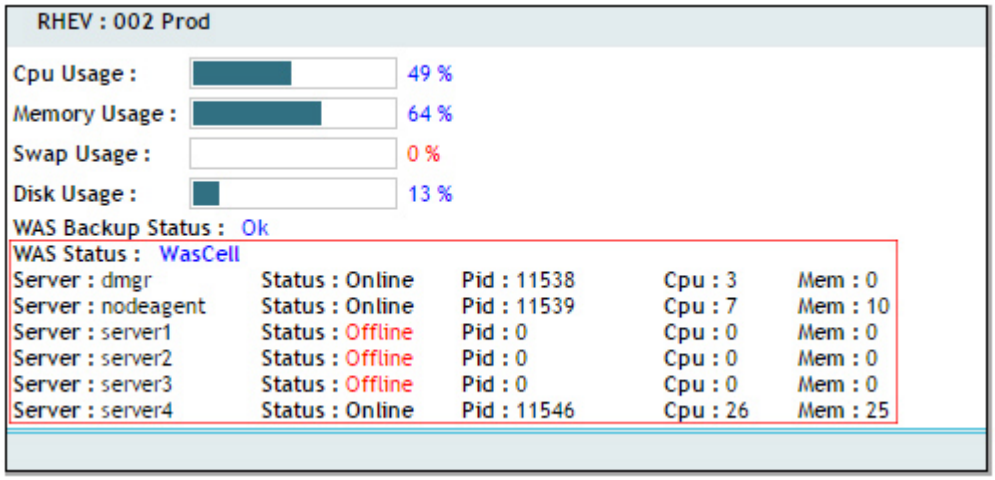
V primeru, da se varnostna kopija ni izvedla oziroma je starejša od enega dneva, spletna aplikacija javi napako (slika 41).



Slika 41: Alarm v primeru, ko se varnostna kopija aplikacijskega strežnika ne izvede

5.2.6 Spremljanje stanja aplikacijskih strežnikov po komponentah

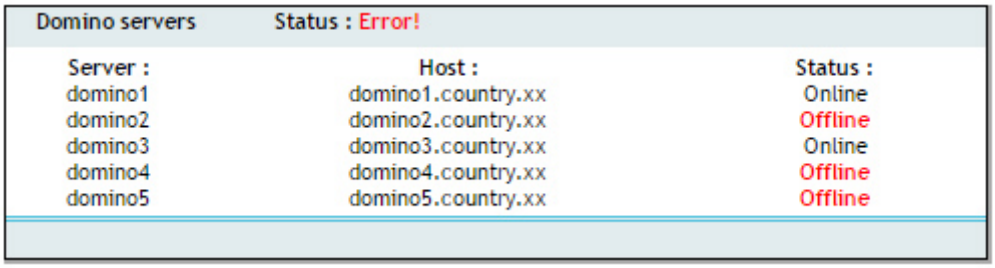
Aplikacijski strežniki tipa IBM WebSphere so sestavljeni iz komponent. Komponente, ki sestavljajo aplikacijski strežnik, so »deployment manager«, »nodeagent« in poljubno število strežnikov. Na vsakem izmed strežnikov tečejo aplikacije. Spletna aplikacija nam omogoča spremljanje, ali komponente delujejo in pa koliko sistemskih virov uporabljajo. Na primer, za posamezno komponento lahko ugotovimo, ali sploh deluje in če, koliko odstotkov centralne procesne enote zaseda ter koliko odstotkov pomnilnika uporablja (slika 42). Spletna aplikacija nam za posamezno komponento pove tudi številko »pid«. To je številka procesa, pod katerim teče komponenta na operacijskem sistemu Linux. Uporabna je v primeru, da moramo proces končati.



Slika 42: Spremljanje delovanja aplikacijskih strežnikov IBM WebSphere na operacijskem sistemu Linux

5.2.7 Spremljanje delovanja strežnikov IBM Lotus Domino

Spletna aplikacija nam omogoča preverjanje delovanja strežnikov IBM Lotus Domino, ki delujejo na operacijskem sistemu Linux. V primeru, da strežnik IBM Lotus Domino ne deluje, nam aplikacija to javi (slika 43). Aplikacija preverja delovanje s pomočjo edinstvenega imena procesa, ki ga uporablja strežnik IBM Lotus Domino na operacijskem sistemu Linux.



Slika 43: Spremljanje delovanja strežnikov IBM Lotus Domino na operacijskem sistemu Linux

6 Nadzorna plošča za aplikacijske strežnike WebSphere

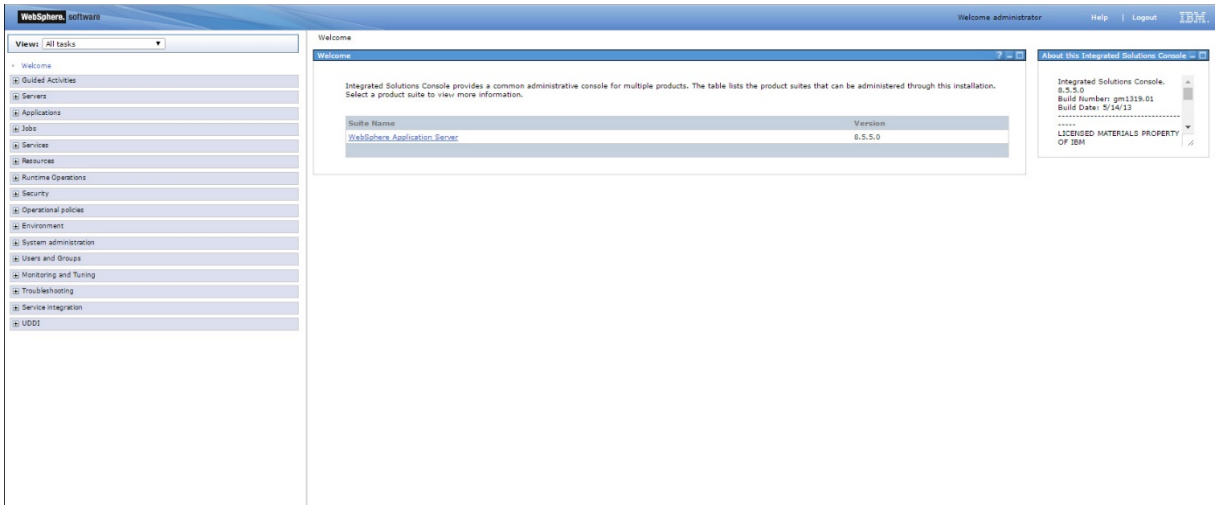
V tem poglavju sta predstavljena aplikacijski strežnik IBM WebSphere in nadzorna plošča spletne aplikacije, ki omogoča nadzor nad komponentami aplikacijskega strežnika IBM WebSphere.

6.1 Aplikacijski strežnik IBM WebSphere

Aplikacijski strežnik IBM WebSphere (slika 44) je komercialna programska oprema, ki opravlja vlogo strežnika spletnih aplikacij. Bolj natančno gre za programsko ogrodje, ki gosti spletne aplikacije, napisane v Javi. Je vodilni izdelek med paleto produktov IBM WebSphere. Njegovi začetki segajo v leto 1998. Trenutna produkcijska verzija je 8.5.

Aplikacijski strežnik IBM WebSphere je zgrajen z uporabo odprtih standardov, kot sta Java EE in XML, in spletnih storitev. Podprt je na naslednjih platformah:

- Windows,
- Linux,
- AIX,
- Solaris,
- IBM i,
- z/OS.



Slika 44: Aplikacijski strežnik IBM WebSphere

6.2 Zgradba in delovanje nadzorne plošče

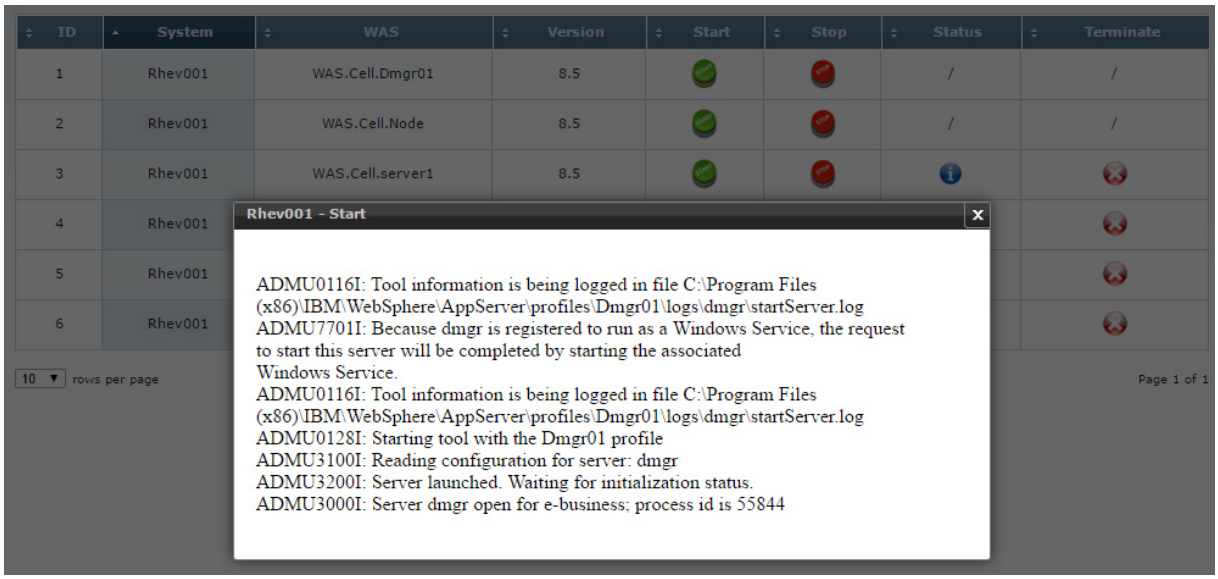
Nadzorna plošča deluje po modelu klient – strežnik (slika 4). Klienta predstavlja majhna aplikacija, ki jo pokličemo preko spletnega strežnika. Aplikaciji podamo podatke o strežniku, geslo, kriptirano v tehnologiji MD5, in ukaz, ki ga želimo izvesti. Strežnik predstavlja samostojno aplikacijo, ki je pognana na vsakem strežniku, na katerem želimo nadzirati aplikacijski strežnik. Aplikacija posluša na določenih vratih in se odziva na ukaze, ki jih pošiljamo iz nadzorne plošče. Aplikacijske strežnike lahko nadziramo na strežnikih z operacijskim sistemom Windows ali Linux. Temu primerno so napisane skripte za zagon, ustavitev, preverjanje stanja in brezpogojno ustavitev komponent aplikacijskega strežnika.

Aplikacijski strežnik je v našem primeru pomemben, saj gosti vse pomembnejše spletne aplikacije. Kot smo že opisali, se tudi ob vseh varnostnih mehanizmih in spremljanju sistema včasih zgodi, da pride do izpada delovanja aplikacij zaradi ene ali več že opisanih težav. V primerih, ko prihaja do težav s podatkovnimi bazami, ki so povezane na več aplikacijskih strežnikih, se pogosto dogaja, da je treba ustaviti in znova zagnati enega ali več aplikacijskih strežnikov ali njihovih komponent. V takih primerih se je treba prijaviti v sistem in zagnati ukaz za ustavitev in zagon aplikacijskega strežnika in njegovih komponent. Če je teh strežnikov več, postane delo časovno zelo zamudno, kar pa je nezaželeno, saj je čas izpada aplikacij zelo pomemben. Nadzorna plošča spletne aplikacije (slika 45) nam delo olajša, saj iz spletne aplikacije lahko nadzorujemo več aplikacijskih strežnikov. Nadzorna plošča omogoča izstavitev naslednjih ukazov za vsako komponento aplikacijskega strežnika:

- start – zagon komponente aplikacijskega strežnika,
- stop – ustavitev komponente aplikacijskega strežnika,
- status – preverjanje, ali je komponenta aplikacijskega strežnika zagnana ali ugasnjena,
- terminate – v primeru, da zaradi določenih težav na sistemu pride do stanja, v katerem aplikacijskega strežnika ni mogoče ugasniti z ukazom stop, uporabimo ukaz terminate, ki zaključi proces, ki poganja komponento aplikacijskega strežnika.

Slika 45: Nadzorna plošča za aplikacijski strežnik IBM WebSphere

V primeru, da poženemo ukaz start, stop, status ali terminate, se odpre novo okno, v katerem dobimo odgovor strežnika (slika 46). V odgovoru so informacije, povezane z ukazom, ki ga izvajamo. Iz informacij je razvidno, ali se je ukaz uspešno izvedel ali ne. V primeru, da se ni, sledi tudi opis napake.



Slika 46: Primer zagona komponente aplikacijskega strežnika IBM WebSphere

7 Sklepne ugotovitve

Cilj diplomskega dela je bil razvoj in predstavitev spletne aplikacije, ki omogoča spremljanje delovanja centralnega računalnika, strežnikov z operacijskim sistemom Linux in nadzor delovanja aplikacijskih strežnikov IBM WebSphere.

Produkcijska verzija spletne aplikacije ustreza vsem zahtevam, ki smo si jih zadali v začetnih fazah razvoja:

- za delovanje sta potrebna samo spletni brskalnik in internetna povezava,
- poenostavljena mora biti do te mere, da je uporabniku dovolj le znanje o uporabi spletnega brskalnika,
- omogočati mora čim enostavnejše dopolnjevanje, dodajanje in odstranjevanje komponent,
- delovati mora v vseh novejših spletnih brskalnikih,
- omogočati mora delo več uporabnikom hkrati.

Spletna aplikacija je namenjena spremljanju parametrov in komponent centralnega računalnika, strežnikov z operacijskim sistemom Linux in aplikacijskih strežnikov. Uporabniku omogoča, da na enostaven način ugotavlja trenutne težave na sistemih, kontrolira delovanje aplikacijskih strežnikov, primerno ukrepa in s tem optimizira odzivne čase ter prepreči delni ali popolni izpad delovanja aplikacij. Uporaba aplikacije je poenostavljena do te mere, da uporabniku ni treba poznati nobenih ukazov za pridobivanje vrednosti parametrov in stanja komponent in ne ukazov za kontrolo delovanja aplikacijskih strežnikov WebSphere.

Pred začetkom in med razvojem je bilo največ pozornosti namenjene izbiri komponent, ki jih bomo spremljali, in načinu predstavitve podatkov, saj je to pri aplikaciji bistvenega pomena. Pri velikem številu podatkov, ki jih je mogoče spremljati, je pomembno, da se spremljajo najbolj uporabni in se predstavijo na način, ki ga je mogoče učinkovito spremljati v spletnem brskalniku.

Zaradi razvoja z uporabo prototipa in dobrega sodelovanja s testnimi uporabniki smo spletni aplikaciji od začetka pa do danes dodali veliko novih parametrov in komponent, ki jih spremljamo. Testni uporabniki so veliko pripomogli k testiranju in izpopolnjevanju alarmov. Spletna aplikacija se občasno, s spremembo določenih sistemov, še vedno nadgrajuje.

Glavni razlog za razvoj spletne aplikacije je bila pohitritev zaznavanja napak in sprememb v hitrosti in stabilnosti delovanja centralnega računalnika, strežnikov z operacijskim sistemom Linux in enostavnejši nadzor aplikacijskih strežnikov IBM WebSphere. Pred razvojem spletne

aplikacije je bil ključen dejavnik pri zaznavanju napak končni uporabnik. Težava pa je v času za pretok informacije o nedelovanju od končnega uporabnika preko službe za pomoč uporabnikov do sistemskega skrbnika. V tem času aplikacije ne delujejo oziroma delno delujejo. Prav to pa je čas, ki se mu sistemski skrbniki želimo izogniti. Z razvojem spletne aplikacije smo dosegli, da lahko opazimo napako pred končnim uporabnikom in s hitrim ukrepanjem preprečimo delni ali popolni izpad aplikacij. V primeru bolj obsežnih težav na sistemih pa s spletno aplikacijo čas izpada močno skrajšamo.

Ocenjujemo, da so cilji uvedbe spletne aplikacije za spremljanje centralnega računalnika, strežnikov z operacijskim sistemom Linux in nadzor nad aplikacijskimi strežniki IBM WebSphere doseženi in da čas, vložen v razvoj in uvedbo spletne aplikacije, ni bil zaman.

Kot vsaka tehnološka zamisel se lahko tudi naša aplikacija v prihodnosti izpopolnjuje. Komponente se lahko dodajajo ali odstranijo glede na potrebe uporabnikov ali spremembe v sami strežniški infrastrukturi. Tako imamo v načrtu naslednje nadgraditve in izboljšave aplikacije:

- posodobitev uporabniškega vmesnika z možnostjo urejanja pravic za dostop,
- razvoj modulov za nadzor novih komponent v strežniških sistemih,
- izpopolnitev alarmov v primeru daljšega izpada komponent.

Seznam slik

Literatura in viri

1. (2016) AJAX. Dostopno na: https://en.wikipedia.org/wiki/Ajax (obiskano 24. maj 2016).
2. M. Ebbers, J. Kettner, W. O'Brien, and B. Ogden, »Introduction to the New Mainframe: z/OS Basics«. IBM Redbooks, 2011.
3. G. F. Gargiulo, »The REXX Language on TSO.« United States, CreateSpace, 2012.
4. (2016) IFL. Dostopno na: http://en.wikipedia.org/wiki/Integrated_Facility_for_Linux (obiskano 24. maj 2016).
5. (2016) IBM. Dostopno na: http://www.ibm.com (obiskano 24. maj 2016).
6. (2016) JavaScript. Dostopno na: https://sl.wikipedia.org/wiki/JavaScript (obiskano 24. maj 2016).
7. (2016) jQuery. Dostopno na: https://en.wikipedia.org/wiki/JQuery (obiskano 24. maj 2016).
8. (2015) Linux. Dostopno na: https://sl.wikipedia.org/wiki/Linux (obiskano 24. maj 2016).
9. (2016) Mainframe computer. Dostopno na: https://en.wikipedia.org/wiki/Mainframe_computer (obiskano 24. maj 2016).
10. (2016) Notepad++. Dostopno na: https://notepad-plus-plus.org/ (obiskano 24. maj 2016).
11. (2016) PHP. Dostopno na: http://www.php.net (obiskano 24. maj 2016).
12. (2016) phpMyAdmin. Dostopno na: https://www.phpmyadmin.net/ (obiskano 24. maj 2016).
13. O. Pomerantz, B. Vander Weele, M. Nelson, and T. Hahn, »Mainframe Basics for Security Professionals: Getting Started with RACF«. United States, Pearson Education, 2013.
14. (2016) zAAP. Dostopno na: https://en.wikipedia.org/wiki/Z_Application_Assist_Processor (obiskano 24. maj 2016).
15. (2016) zEnterprise System. Dostopno na: https://en.wikipedia.org/wiki/IBM_zEnterprise_System (obiskano 24. maj 2016).
16. (2016) zIIP. Dostopno na: https://en.wikipedia.org/wiki/ZIIP (obiskano 24. maj 2016).
17. (2016) z/OS. Dostopno na: https://en.wikipedia.org/wiki/Z/OS (obiskano 24. maj 2016).